

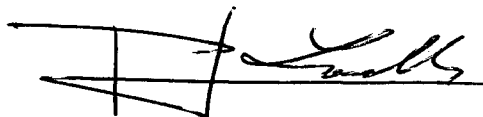
SIMULAÇÃO E REALIZAÇÃO DE UM COMPUTADOR COM

BASE NUM PROCESSADOR MONOLÍTICO

Christian Lenz Cesar

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE
PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE
JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO
DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.).

Aprovada por:



Presidente



RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

OUTUBRO DE 1973

A G R A D E C I M E N T O S

Meus sinceros agradecimentos ao coordenador do Programa de Engenharia de Sistemas e Computação, professor Nelson Maculan, pelo apoio e ao professor Celso de Renna e Souza por ter tornado realidade a parte prática deste trabalho.

R E S U M O

O presente trabalho descreve a arquitetura de um pequeno computador realizado com base num processador monolítico e sua simulação.

Primeiramente estabelece-se os circuitos externos ao processador necessários ao seu funcionamento.

Em seguida mostra-se um simulador que executa instruções dadas em sua forma binária. O usuário pode definir dentro do simulador a organização da memória, dos periféricos e do sistema de interrupção.

Finalmente descreve-se a implementação do computador.

A B S T R A C T

This paper describes an architecture for a small computer using a microprocessor as its central processing unit.

In the first part of the work it is established the necessary external logic circuits for the microprocessor.

The second part presents a simulator that executes instructions given in binary form. The user may define the memory, peripherals and interrupt system organization inside the simulator.

Finally it is described the implementation of the computer.

Í N D I C E

Capítulos:	Páginas:
APRESENTAÇÃO	1
 I O MICROCOMPUTADOR	
I.1. Introdução	4
I.2. Processador	6
I.3. Memória	12
I.4. Entrada e Saída	15
I.5. Sistema de Interrupção	18
I.6. Conclusão	21
 II O SIMULADOR	
II.1. Introdução	25
II.2. Programa Principal	28
II.2.1 Módulos	28
II.2.2 Organização da Memória	32
II.2.3 Interrupções e	
Entradas/Saídas	35
II.3. Utilização do Simulador	41
II.3.1 Cartões de controle e	
programa	42
II.3.2 Interpretação dos resulta-	
dos da simulação	45
II.4. Conclusão	47

Capítulos:

Páginas:

III	A REALIZAÇÃO	
III.1.	Introdução	50
III.2.	Projeto do microcomputador	52
III.2.1	O painel	53
III.2.2	Tecclas	57
III.2.3	Matriz Programável	59
III.3.	Implementação	62
III.3.1	Organização do protótipo	63
III.3.2	Programas de painel	67
III.4	Conclusão	72
CONCLUSÃO		78
BIBLIOGRAFIA :.....		80
Apêndices:		
A	CIRCUITOS	81
B	COMO USAR O SIMULADOR	98
C	LISTAGENS	103

A P R E S E N T A Ç Ã O

A integração de funções lógicas no início da década de 60 libertou o projetista de circuitos do trabalho de montagem dessas funções com componentes discretos. Desde então a eletrônica digital teve um desenvolvimento espantoso com a multiplicação das tecnologias de fabricação dos circuitos, cada uma trazendo novos progressos em termos de velocidade e de densidade de integração.

A integração em larga escala (LSI-Large Scale Integration), apanágio da tecnologia MOS (Metal Oxide Semiconductor), permitiu a colocação num só circuito integrado de funções cada vez mais complexas, como por exemplo, memórias a semicondutores de grande capacidade.

Foi a partir dessa tecnologia que surgiu o microprocessador, que consiste num conjunto de circuitos LSI, cada um contendo uma parte importante de um computador digital, por exemplo toda a unidade aritmética e lógica ou mesmo a unidade central de processamento (CPU - Central Processing Unit).

O impacto desses microprocessadores se verifica especialmente em áreas antes dominadas pelo minicomputador e pelo sistema digital feito "sob medida" (special-purpose), tais como controle de processo, controle de máquina, periféricos

de computador, máquinas de calcular, instrumentação, terminais inteligentes e em comunicações, por serem de baixo custo e tamanho reduzido. Os sistemas de processamento montados com êsses novos componentes foram chamados de microcomputadores.

Lançado em 1972 o INTEL 8008 foi um dos primeiros microprocessadores vendidos no mercado. Constituído por apenas um circuito integrado de 18 pinos, é uma CPU que trabalha em paralelo sobre palavras de oito bits, possui um repertório de 48 instruções e endereça até 16K palavras de memória.

Este trabalho resulta do estudo aprofundado da 8008 e do microcomputador SIM8-01 da mesma companhia e tem por objetivos:

- dar ao projetista de circuitos lógicos regras e modelos para a implementação de um sistema digital baseado na 8008
- criar um suporte de programação para o projetista e usuário da 8008 a fim de facilitar a implementação e correção dos programas a serem colocados em ROM (Read Only Memory) e outras memórias

O trabalho está dividido em três capítulos. Inicialmente propõe-se uma arquitetura para um microcomputador que usa a 8008 indicando os sinais de controle necessários ao seu funcionamento bem como a maneira de gerá-los por circuito. Al-

gumas regras são estabelecidas para o projeto da memória, das entradas e saídas, e do sistema de interrupção.

No segundo capítulo descreve-se um programa em PL/1 que simula a 8008 permitindo ao usuário a definição da organização da memória, das entradas e saídas, e das interrupções, que compoem o microcomputador.

O terceiro capítulo trata da implementação real de um microcomputador usando o modelo da primeiro capítulo.

É importante acentuar que a compreensão destes capítulos depende da leitura prévia do manual da 8008, referencia básica de toda a tese. Não se pretendeu que êste trabalho fosse outro resumo da 8008. Apenas os aspectos menos claros do manual é que serão tratados nas discussões.

C A P Í T U L O I

O MICROCOMPUTADOR

I.1 - INTRODUÇÃO

A arquitetura a ser descrita para o microcomputador, resultou da experiência adquirida nos últimos anos em nossos laboratórios de circuitos digitais e da presente (1972) situação do mercado brasileiro de eletrônica.

A quase totalidade dos circuitos digitais encontrados nos revendedores das grandes cidades são funções lógicas simples. Raramente se encontram circuitos LSI. Por essas razões é que o chaveamento de informações que convergem para um mesmo ponto é tradicionalmente feito utilizando-se circuitos com saída em coletor aberto, ao contrário dos americanos que podem se dar o luxo de usar multiplexadores ou lógica a três estados.

A arquitetura repousa portanto sôbre uma técnica já bem dominada que é a de barras de coletor aberto. Esta escolha resultou em que apenas a 8008, a memória e os inversores de baixa potencia, necessários ao "interface" MOS-TTL, fossem buscados fora do Brasil.

A descrição que se segue trata os circuitos do ponto de

vista estritamente lógico. Somente nas conclusões deste capítulo é que serão discutidos os problemas práticos que poderão ocorrer na implementação.

A fim de evitar confusão usar-se-á a seguinte convenção:

microprocessador - a 8008

processador - a 8008 e o conjunto de circuitos necessários ao seu funcionamento

microcomputador - o processador com a memória, entradas e saídas, e o sistema de interrupção

I.2 - PROCESSADOR

A organização de um processador que usa a 8008 é a da figura 1.1. A barra de dados admite fluxo de informação em ambos os sentidos. As informações que saem da 8008 são colocadas na barra de memória, ou barra M, que é a continuação lógica da barra de dados. Ao contrário, a barra de entrada, ou barra E, por onde chegam as informações, está isolada por meio de porta lógica que será amostrada em T3A, quando a 8008 estiver pronta para receber o octeto de bits. Os registros RL e RH armazenam durante os ciclos PCI, PCR e PCW, as partes baixa e alta do endereço, respectivamente. Para o ciclo de entrada e saída, (PCC), RL alimenta a barra de saída, ou barra S, e RH contém o número do periférico.

Os sinais de controle da 8008 - "status", "sync", CCl e CCo (os dois últimos obtido através de RH) - e a fase ϕ_2 do relógio mestre são utilizados para a geração de todos os sinais necessários ao sistema (controle de escrita na memória, sincronização da interrupção, estados, ciclos, controle de multiplexagem na barra de entrada, etc.).

As figuras 1.2 e 1.3 mostram os circuitos de geração desses sinais e a figura 1.4 um diagrama de tempo que indica o relacionamento entre os sinais mais importantes:

ϕ_{22} - a quarta fase do relógio. É o sinal mais impor-

tante, sendo usado para a obtenção de quase todos os outros.

STROBE LO - carrega a palavra presente na barra M em RL

STROBE HI - carrega a palavra presente na barra M em RH

PCI,PCR,PCW,PCC - os ciclos do processador, obtidos pela decodificação de CCO e CCl.

T1,T2,T3,T4,T5,T1I,WAIT,STOP - os estados do processador, obtidos pela decodificação de S0, S1 e S2.

R/W - comando de escrita em memória do dado presente na barra M. Só ocorre durante o ciclo PCW.

T3A - amostra o dado presente na barra E quando a 8008 está recebendo informações.

Esses sinais serão utilizados na obtenção dos sinais de multiplexagem das três entidades básicas da máquina ligadas à barra E (memória, periféricos de entrada e sistema de interrupção) e dos sinais de chamada a periféricos de saída ligados à barra S. Serão chamados de LIB (libere) no caso da barra de entrada e CARR (carregue) no caso da barra de saída. Os LIB são tais que apenas um poderá ocorrer (nível 1 lógico) num dado instante. Os CARR não possuem esta restrição, porém na prática, serão também mutuamente exclusivos. Isto é devido ao fato que a barra E é do tipo coletor aberto (as barras S e M não são do tipo coletor aberto).

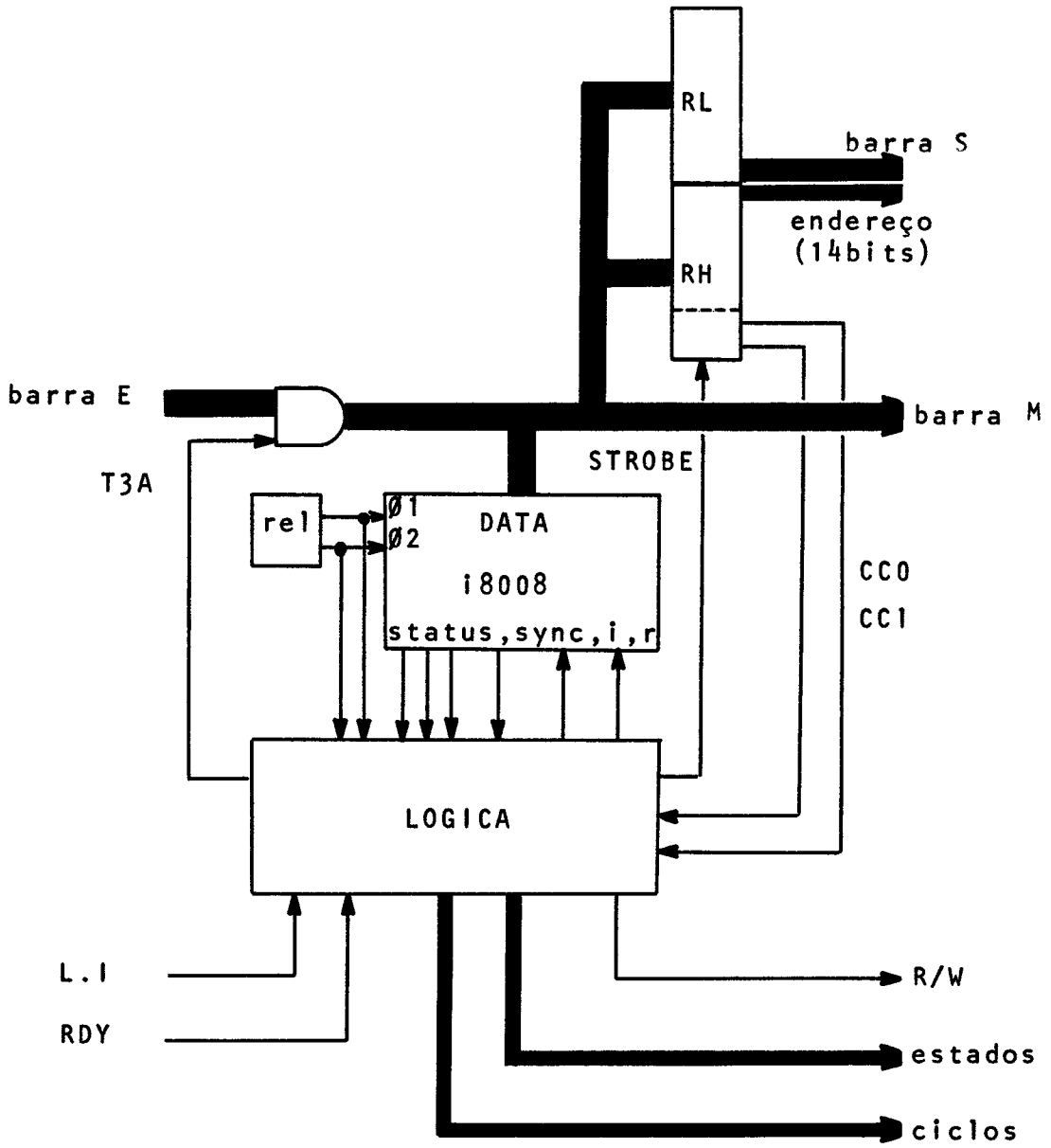


fig. 1.1

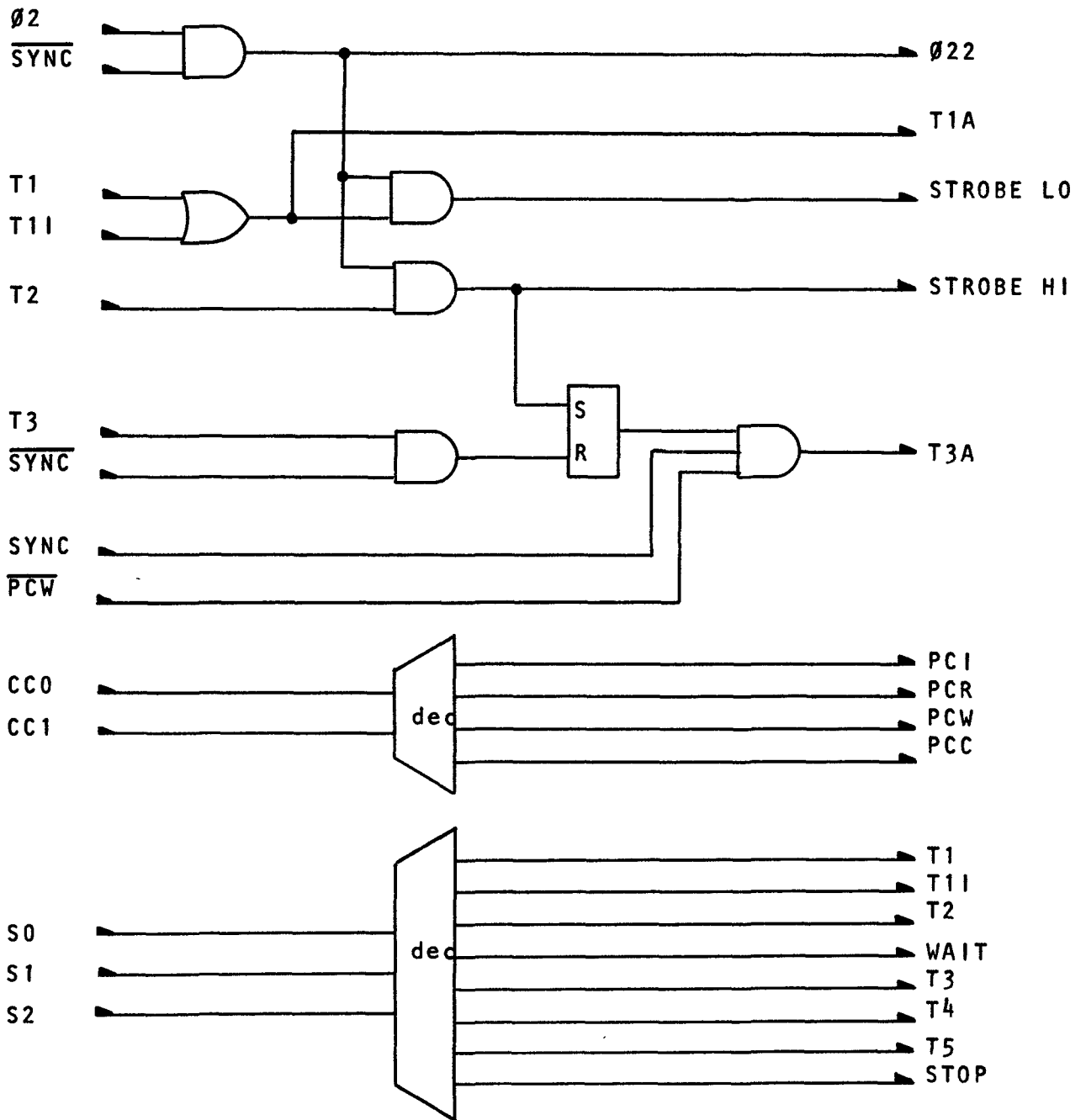


fig. 1.2

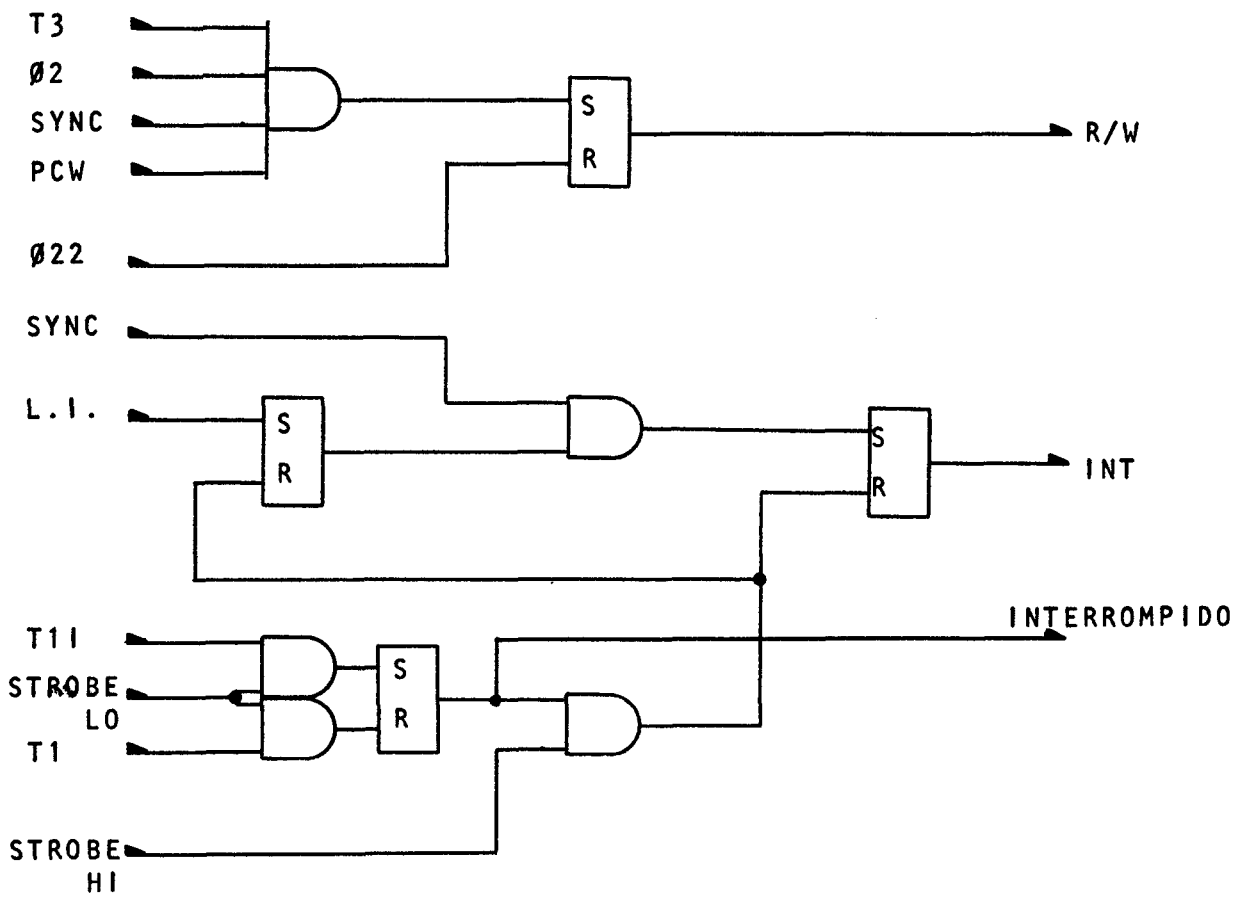


fig. 1.3

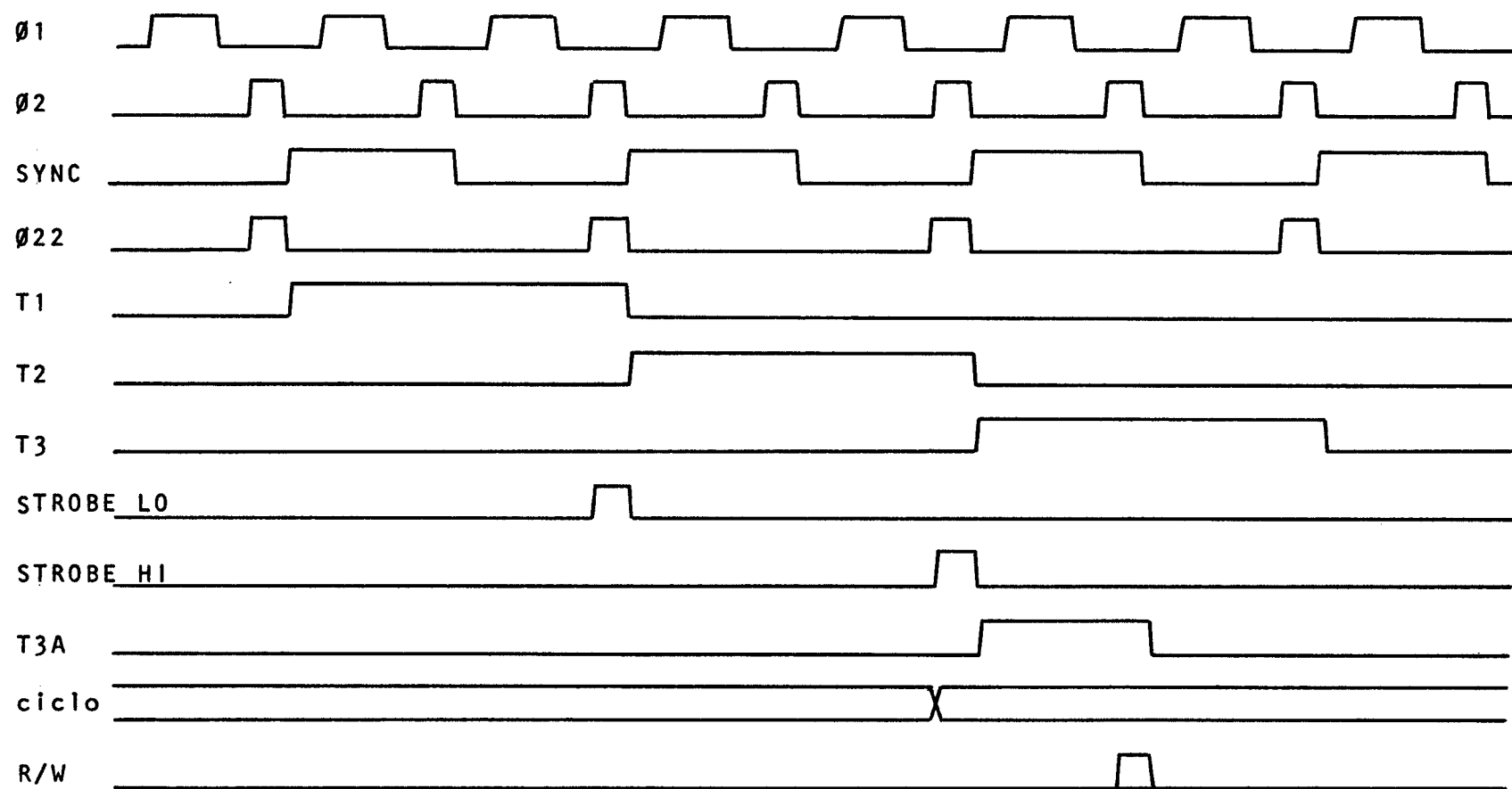


fig. 1.4

I.3 - MEMÓRIA

A memória principal do microcomputador é organizada segundo as necessidades do usuário e geralmente será do tipo a semicondutor. Normalmente haverá uma divisão por módulos (considera-se módulo de memória aquela que já tiver circuito próprio de decodificação de endereço) com a parte de ROM nos endereços mais baixos. A decodificação da parte alta do endereço é função do tamanho dos módulos de memória utilizados; a parte alta do endereço é usada para selecionar o módulo. No caso de uma subdivisão uniforme da memória, ie, módulos de mesmo tamanho, a organização será a da figura 1.5.

A palavra lida em memória só será colocada na barra E durante LIB MEM, já que todos os módulos são isolados desta barra por uma porta lógica (figura 1.6). No caso de RAM (Random Access Memory) o dado a ser escrito está na barra M.

O sinal LIB MEM existirá quando se verificarem as seguintes condições:

- ciclos PCI ou PCR
- processador não interrompido

No caso de memórias lentas há a necessidade de sincronização do processador a elas. Para isto existe o sinal de RDY que poderá estar em 1 ou 0 lógicos, significando que o processador tende a passar de T2 a T3 diretamente ou tende a entrar

no estado de WAIT após T2, respectivamente. No primeiro caso, o módulo de memória lenta deverá forçar RDY a 0, inibindo assim a passagem de T2 para T3, e mantê-lo nesse nível até o fim da operação de leitura ou escrita. No segundo caso, RDY deverá ser levado a 1 quando o acesso tiver sido terminado, fazendo com que o processador deixe o estado de WAIT passando a T3.

A capacidade de memória pode ser aumentada por meio de instruções de entrada e saída. Pode-se concatenar registros de 8 bits aos 14 bits já existentes, carregando-os por meio de instruções OUT. Isto permite uma expansão da memória praticamente ilimitada.

endereço

RH □ RL

(14 bits)

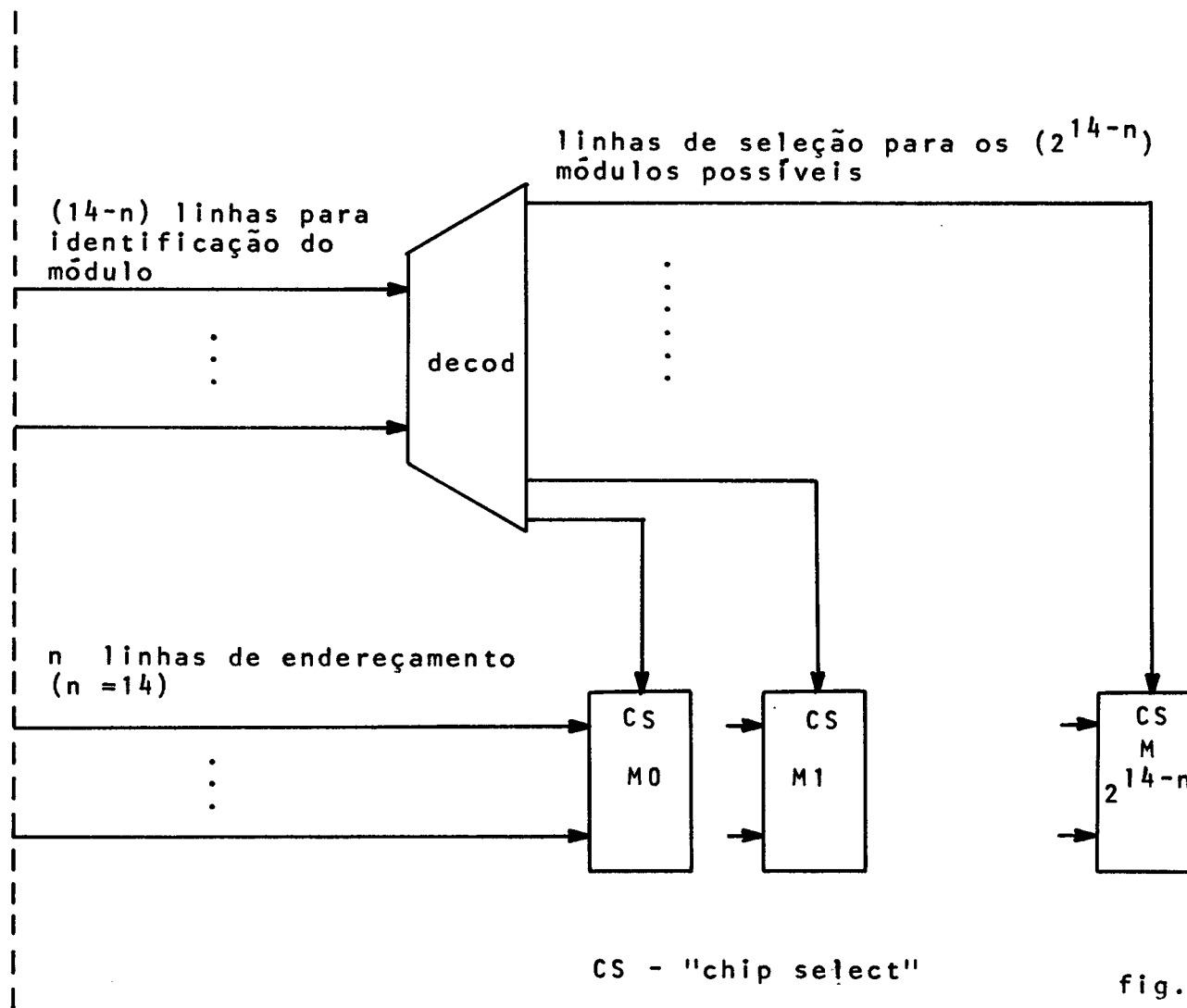


fig. 1.5

I.4 - ENTRADA E SAIDA

Toda comunicação de entrada e saída é feita durante o ciclo PCC. Os periféricos de entrada estão ligados à barra E através de portas lógicas que serão amostradas pelos sinais LIB INP correspondentes. Os periféricos de saída ligados à barra S serão carregados pelos sinais CARR OUT correspondentes (figura 1.6).

O sinal LIB INP 'número do periférico', existirá quando se verificarem as seguintes condições:

- ciclo PCC
- chamada ao periférico de entrada específico (resulta da decodificação do código contido na instrução INP)

O sinal CARR OUT 'número do periférico', existirá quando se verificarem as seguintes condições:

- ciclo PCC
- chamada ao periférico de saída específico (resulta da decodificação do código contido na instrução OUT)

A decodificação do código de periférico, presente em RH, tem duas soluções extremas: total decodificação do código junto ao processador com saída de linhas individuais ou decodificação local, ie, no próprio periférico. A escolha de uma solução entre esses dois extremos dependerá do número e tipo

dos periféricos, bem como da distância entre êles e o processador.

Como no caso da memória, o processador pode ser sincronizado com os periféricos. As explicações dadas anteriormente sobre o sinal de RDY são válidas aqui.

Para os periféricos de entrada, numerados de 0 a 7, a palavra presente na barra S pode ser usada como comando. O dado a ser enviado ao processador será colocado na barra E. Para os periféricos de saída, numerados na base octal de 10 a 37, o dado está na barra S, enquanto que a barra E permanecerá neutra.

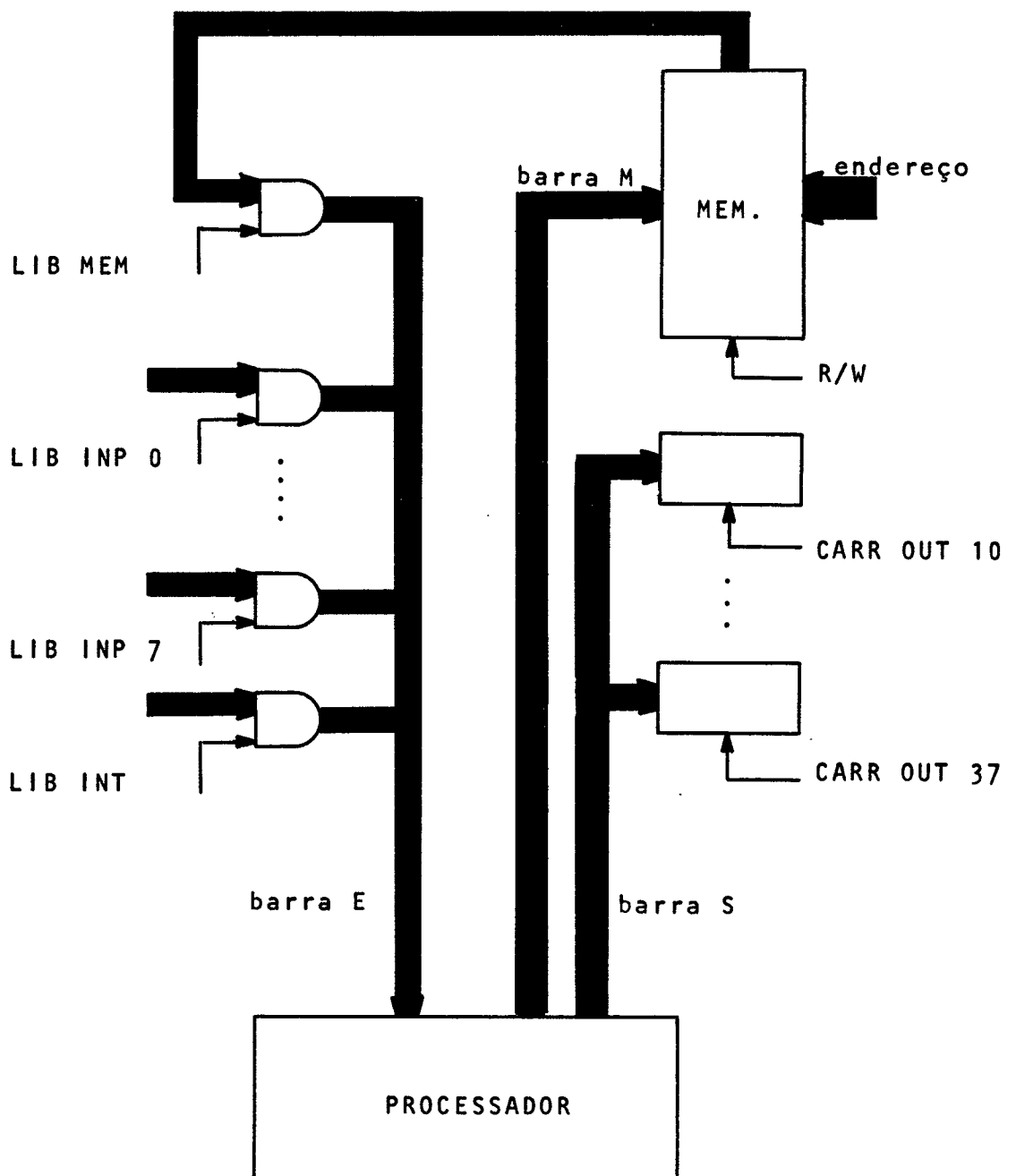


fig. 1.6

I.5 - SISTEMA DE INTERRUPTÃO

Para o projeto do sistema de interrupção dispõe-se de uma linha de interrupção (L.I.), por onde será enviado o sinal que irá interromper a 8008, e a barra E onde será colocada a instrução. A ligação a essa barra é feita através de porta lógica (figura 1.6), amostrada por LIB INT, que ocorre nas seguintes condições:

- ciclo PCI, ou PCR no caso de instruções longas (duas ou três palavras)
- quando o processador estiver interrompido

Em alguns casos, tipicamente quando a instrução for longa, poderá haver a necessidade de mais outra condição, além das duas citadas acima, gerada pelo próprio sistema de interrupção, a fim de diferenciar cada octeto da instrução longa.

Um sinal poderá ser colocado em L.I. sempre que a última interrupção tiver sido processada (uma interrupção é considerada já processada durante ou após os estados T3, T4 ou T5 do último ciclo da instrução relativa a esta interrupção). Os sinais que não respeitarem este limite serão ignorados pelo processador.

A primeira palavra da instrução deve estar presente na barra E durante o primeiro ciclo PCI após a colocação do sinal em L.I.. Os eventuais endereços ou dados imediato (caso das ins-

truções longas) deverão ser colocados a sua vez na barra E nos sucessivos ciclos PCR. A figura 1.7 ilustra a sequência de eventos.

Em termos da 8008 há a necessidade de sincronização dos sinais de interrupção, que poderão chegar aleatoriamente no tempo. Essa função será atribuída a um circuito sequencial que armazenará o sinal (de qualquer largura) enviando-o no momento certo para a 8008.

O reconhecimento da interrupção pela 8008 implica na substituição de T1 por T1I em todos os ciclos da instrução (isto é válido para qualquer uma das 48 instruções da 8008). Além disso, o processador fornecerá um sinal (INTERROMPIDO) que permanecerá ligado durante todo o processamento da interrupção.

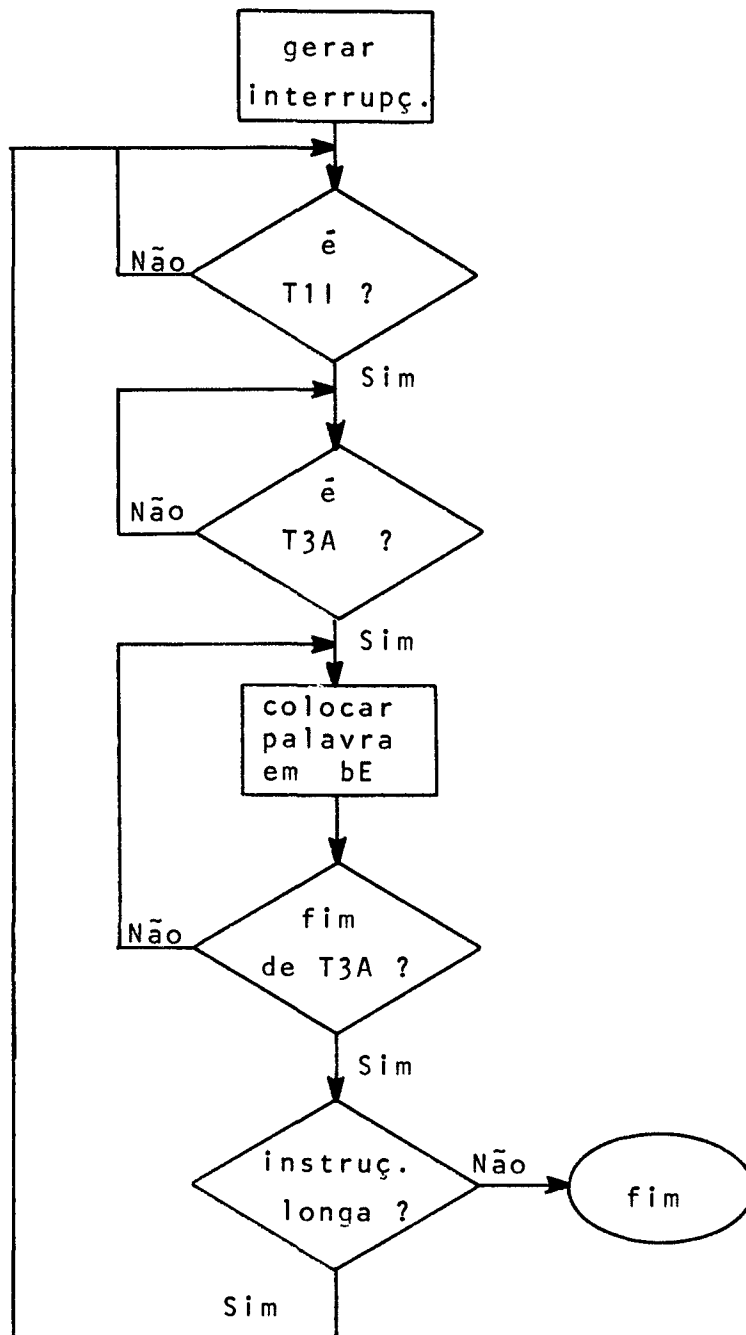


fig 1.7

I.6 - CONCLUSÃO

A utilização da barra do tipo coletor aberto com a 8008 oferece como vantagem a modularidade, ie, a colocação de um novo circuito sôbre a barra E não implica em alterações no sistema original. A desvantagem é a limitação inerente ao coletor aberto quanto ao número de circuitos que podem ser conectados à barra (normalmente 25 para o integrado 7401 - 2 input positive nand gate with open-collector output - quando se garantir que apenas um circuito estará ativo de cada vez e que a barra termina numa porta lógica com FAN-IN igual a 1). Por isso é aconselhável ligar as saídas de todos os módulos de memória na entrada de uma só porta lógica com saída coletor aberto (figura 1.6). Isto, e o fato de só existirem no máximo 8 periféricos de entrada, permitirá ao sistema de interrupção o acesso à barra E através de várias portas lógicas, cada qual com seu sinal de amostragem LIB INT.

A barra S não precisa ser do tipo coletor aberto, porém, como nela estarão ligados todos os periféricos de saída e todo o endereçamento da memória, é absolutamente necessário que o registro RL tenha alto FAN-OUT.

Cuidado também deve ser exercido na barra M, pois a soma dos FAN-IN dos módulos de memória (na entrada de dados) não deve ultrapassar o FAN-OUT do circuito que a alimenta.

O FAN-OUT normal de 10 (família TTL) para o RH geralmente será suficiente.

Os esquemas apresentados para o processador foram projetados para o caso geral de utilização da 8008. Em aplicações particulares, alguns circuitos poderão ser eliminados. A sincronização do sinal de interrupção, por exemplo, poderá ser desnecessária se o sinal em L.I. tiver largura suficiente para interromper a 8008. Também, o circuito que gera R/W só é necessário em sistemas que utilizem memória RAM.

A organização da memória merece cuidados especiais devido ao carregamento em tempos diferentes dos registros RL e RH, e ao fato que nas memórias MOS o tempo de acesso após a mudança das linhas de endereçamento (supondo o módulo de memória já selecionado) é bem maior que o tempo de acesso após a seleção do módulo de memória (supondo o endereço já presente nas linhas de endereçamento). Portanto, deve-se primeiramente gerar os níveis para as linhas de endereçamento e depois o sinal que seleciona o módulo. Isso pode ser realizado com o processador, se os bits de endereçamento estiverem totalmente contidos em RL, quando então as linhas terão tempo (entre STROBE LO e STROBE HI) para se estabilizar até a seleção do módulo de memória, obtida pela decodificação (que é rápida no caso) dos bits presentes em RH, e mais aqueles de RL que não participam da ativação das linhas de endereçamento. Isto implica num tamanho ótimo do módulo de memória

de 256 palavras ($n=8$ na figura 1.5), já que a utilização de módulos menores representa um aumento na área ocupada pela memória.

A utilização, nas instruções INP, do valor do acumulador, é particularmente útil para a leitura de condições relativas a um periférico ou para comandos especiais. Por exemplo, pode-se saber se uma fita está ocupada, comandar o enrolamento rápido da fita, etc. Isto é feito precedendo a instrução de INP de um LAI que carrega um código que será interpretado pelo periférico, e eventualmente introduzindo uma instrução de teste sobre o octeto lido após a instrução de INP.

A existência de diversas partes do sistema com capacidade de interrupção cria o problema da coincidência de sinais em L.I.. Um esquema de prioridade deverá ser estabelecido, se necessário, entre os elementos capazes de interromper a 8008; pode ser uma simples fila (FIFO) ao lado do processador, que armazena as instruções que chegam para tratamento sob interrupção, ou alguma solução mais complexa que envolva uma hierarquia entre esses elementos.

Uma das particularidades do processador é o fato de não fazer distinção durante o ciclo PCR entre instruções imediatas e de referencia memória, ie, RL e RH não sabem se o endereço recebido no ciclo de leitura é proveniente do PC ou de H concatenado com L (os dois últimos registros da 8008).

Durante o processamento normal isto não causa problema, pois é uma leitura na memória que é envolvida nos dois casos. Mas sob interrupção implica que se deve optar durante PCR, ou pela leitura em memória (caso da instrução curta LrM), ou pela leitura do sistema de interrupção (como é o caso de instruções longas do tipo LrI). A primeira opção é inadequada, pois não permite instruções de chamada a subrotina, que, como as instruções de referência imediata, são instruções longas (o sistema de interrupção enviaria à 8008 o primeiro octeto da instrução CAL durante PCI, e depois a 8008 leria a memória, endereçada pelo PC, para obter o endereço da subrotina, com resultados imprevisíveis). Só resta a opção de se restringir as instruções que podem ser resolvidas sob interrupção ao conjunto que não inclui instruções de referência memória com leitura (a escrita é permitida, pois acontece no ciclo PCW). Isto é perfeitamente tolerável para a maioria dos casos. Instruções associadas a interrupções são tipicamente CALL, RESTART e HALT. A eliminação das instruções LrM do repertório daquelas que podem ser resolvidas sob interrupção poderia ter sido evitado pelos engenheiros da INTEL se durante a resolução da instrução LrM o ciclo PCR aparecesse com T1 em vez de T1I. Infelizmente este não é o caso.

C A P Í T U L O I I

O SIMULADOR

II.1 - INTRODUÇÃO

O simulador escrito em PL/1 tem como finalidade o desenvolvimento e correção de programas para o microcomputador. Uma sequencia de instruções em linguagem de máquina da 8008 é fornecida ao simulador que lista como resposta o estado da 8008 a cada passo da resolução do programa.

Entradas e saidas, e interrupções podem ser definidos pelo programador dentro do programa principal em posições pre-estabelecidas.

O estabelecimento da imagem da memória é feita por meio de cartões de controle colocados antes dos dados.

Como outra opção pode-se controlar a impressão dos resultados informando ao simulador quais as instruções que aparecerão na listagem final.

Neste capítulo, ao contrário do primeiro, dividir-se-á a memória em páginas. Em princípio êste novo conceito e módulo de memória se confundem; um módulo seria uma página. Mas se no projeto do computador não se pode abstrair dos tamanhos dos módulos por causa da decodificação, no simulador se fará uma

divisão uniforme do vetor memória a fim de facilitar a programação PL/1. As páginas maiores serão portanto subdivididas em menores com consequencia que um módulo da máquina real corresponderá a uma ou mais páginas dentro do simulador.

Os tempos envolvidos na simulação, como o de instrução, de acesso à memória e outros, serão medidos com a unidade de tempo do simulador - o período do sinal de sincronismo gerado pela 8008 (entre 2 e 3 microsegundos). Não serão admitidos tempos com parte fracionária, devendo normalmente se arredondar o valor para o inteiro imediatamente superior.

Inicialmente descreve-se as características principais do simulador e em seguida a sua utilização. A listagem do programa principal se encontra no apêndice C .

A fim de evitar confusão usar-se-á a seguinte convenção:

programa principal ou simulador - lista de instruções

PL/1 que realizam a simulação do microcomputador.

vetor memória - vetor de 16384 palavras de 8 bits uti-

lizado como memória pelo microcomputador simulado.

Este vetor é subdividido em páginas.

programa - lista de instruções 8008 armazenada na memória.

imagem da memória - máscara do vetor memória para a definição de

- .tipo de cada página, ie, se é ROM ou RAM
 - .tempos de acesso, leitura ou escrita, a cada página
 - .comportamento do sistema no caso de endereçamento de página inexistente
- ponto de quebra - par de endereços que definem o início e fim da impressão dos resultados.

II.2 - PROGRAMA PRINCIPAL

A primeira função importante do simulador é a medição do "tempo real" de processamento da 8008. Isto é feito através da variável `NUMERO_DE_ESTADOS`.

Inicializada em zero, `NUMERO_DE_ESTADOS` será incrementada a medida que instruções forem resolvidas (tempos fixos dentro do programa principal) e a cada chamada à memória, às entradas e saídas, e ao sistema de interrupção (tempos definidos pelo usuário). A introdução desses tempos no programa principal se faz com instruções PL/1 do tipo

```
NUMERO_DE_ESTADOS = NUMERO_DE_ESTADOS + tempo ;
```

II.2.1 - Módulos

O programa principal é subdividido em módulos que consistem numa série de instruções PL/1 que realizam uma função específica dentro do simulador. Há dois conjuntos de módulos: os processados uma só vez e os utilizados repetidas vezes durante o processamento.

O primeiro conjunto, que faz a inicialização do simulador, é formado pelos seguintes módulos:

módulo de declaração e inicialização de variáveis

nêle estão declaradas tôdas as variáveis utili-

zadas dentro do programa principal, bem como eventuais valores iniciais.

módulo de leitura dos cartões de controle

leitura dos cartões de controle fornecidos pelo usuário para o estabelecimento da imagem da memória e pontos de quebra dentro do programa.

módulo de carregamento dos programas

os programas do usuário são carregados em posições do vetor memória por êle definidas, conforme a imagem de memória anteriormente estabelecida.

No outro conjunto encontram-se os módulos que simulam o funcionamento da 8008: interrupção, ciclos da máquina, decodificação e resolução das instruções (a uma instrução pode corresponder mais de um módulo).

As figuras 2.1 e 2.2 mostram a sequencia dentro do simulador para o processamento de cada instrução do programa, com ou sem interrupção.

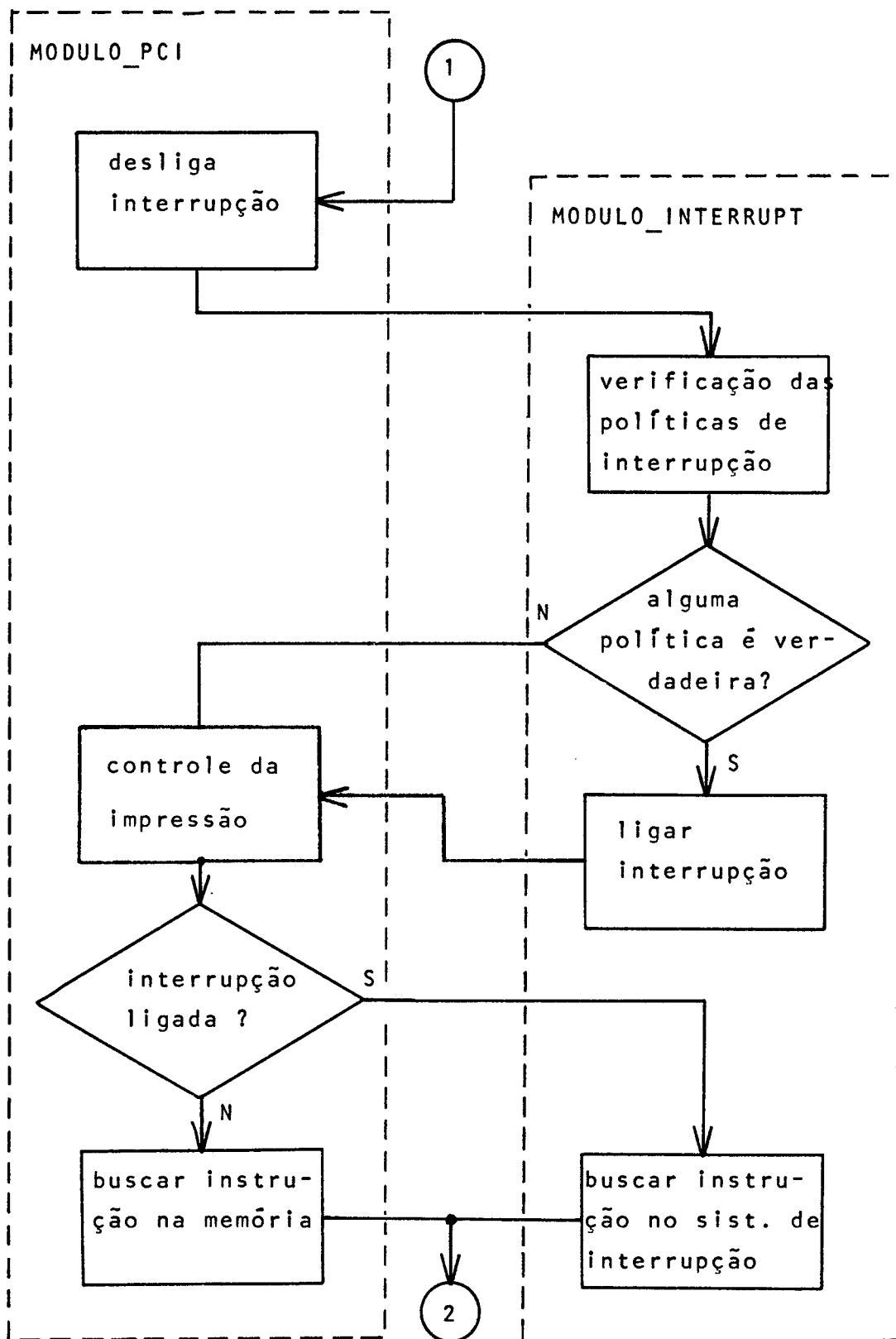


fig 2.1

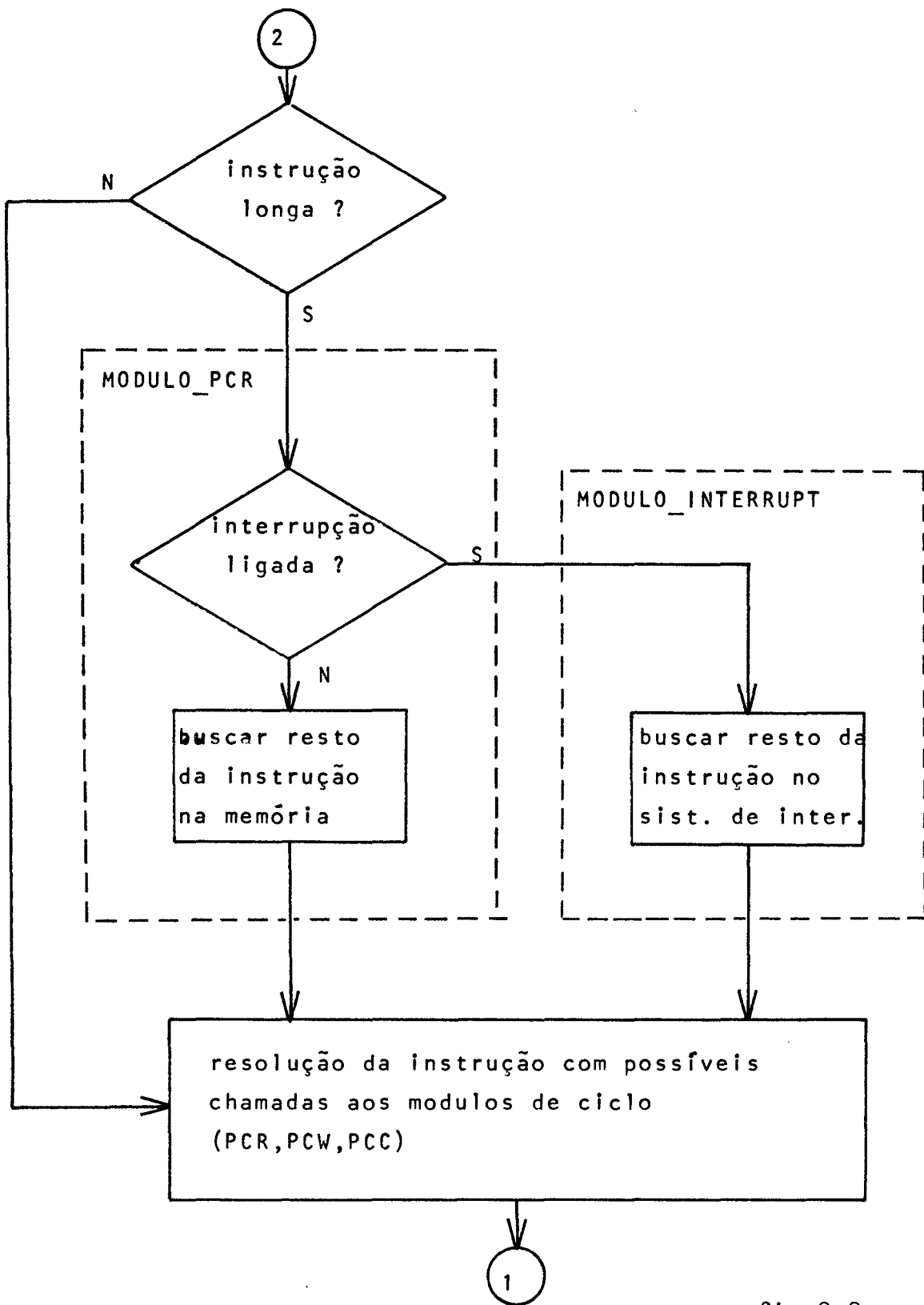


fig 2.2

II.2.2 - Organização da memória

A organização da memória do microcomputador simulado será definida por três variáveis dentro do programa principal:

MEM - o vetor memória de 16384 palavras de 8 bits

PAG - o tamanho da página

IMAGEM_MEMORIA - um vetor estrutura de 16384/PAG posições, ie, o número de páginas que podem existir

Os ramos da estrutura IMAGEM_MEMORIA são TIPO (indica se a página é ROM, RAM, ...), CARREGAMENTO (indica se a página existe e portanto passível de ser carregada com os programas pelo módulo de carregamento), TEMPO_LEITURA e TEMPO_ESCRITA.

Estabelecido o valor de PAG, o programa principal montará uma estrutura como a da figura 2.3 (é apenas um exemplo), ie, cada elemento de IMAGEM_MEMORIA apontando para uma página (direta ou indiretamente) e a definindo. Cada posição do vetor conterá uma das seguintes configurações:

ROM 1 t1 te - é uma página ROM com tempos de acesso t1 e te

RAM 1 t1 te - é uma página RAM com tempos de acesso t1 e te

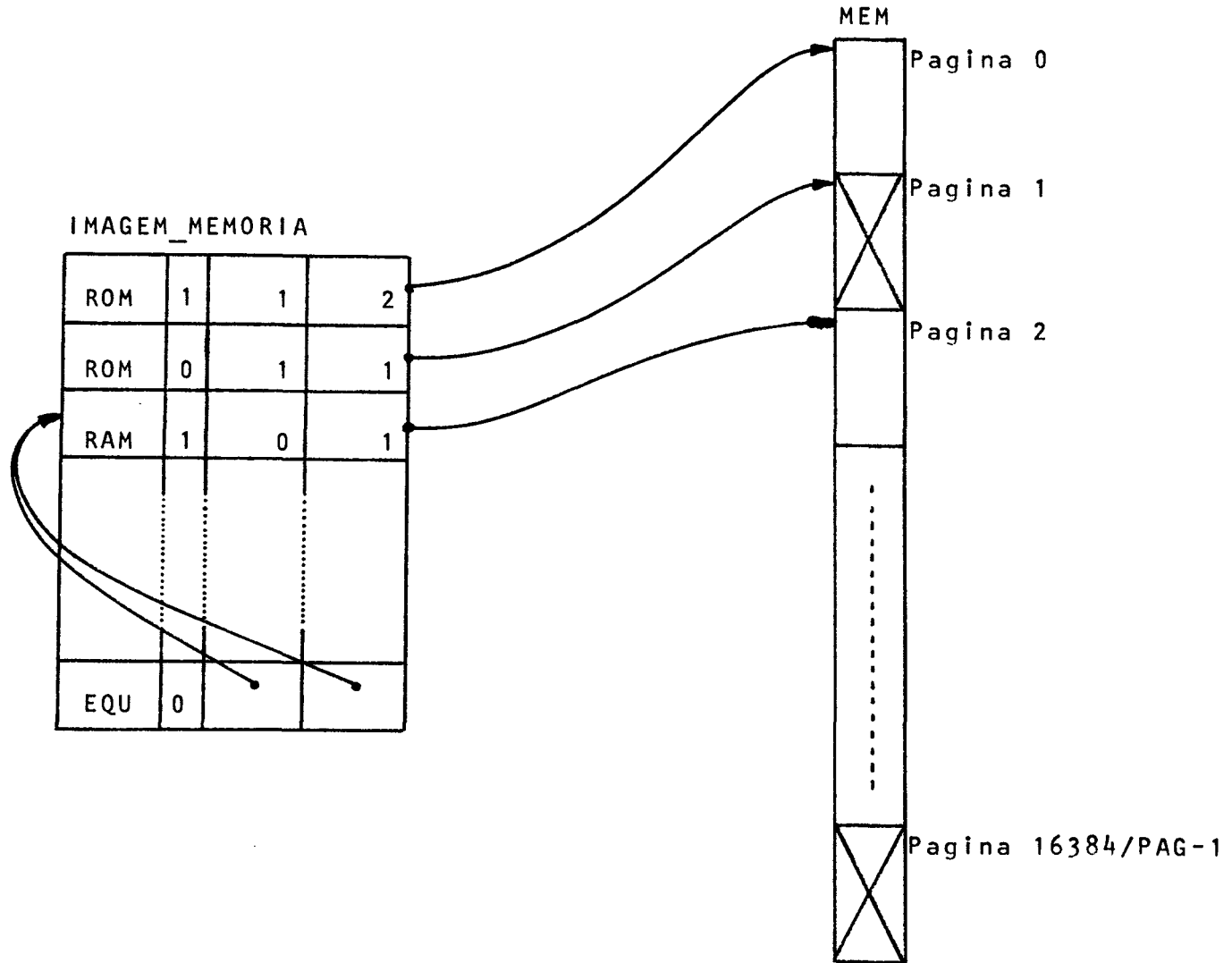
ROM 0 t1 te - é uma pagina inexistente. Quando endereçada pelo programa funciona como ROM com tempos de acesso t1 e te.

EQU O t1 te - é uma página inexistente. Quando endereçada pelo programa, equivale a chamar uma página existente. t1 e te servem como apontadores indiretos para esta página.

~~xxx~~ O t1 te - é uma página inexistente. Quando endereçada pelo programa, a simulação é cancelada. O conteúdo de TEMPO_LEITURA e TEMPO_ESCRITA é indeterminado.

Para cada acesso à memória haverá o estabelecimento do endereço (END_MEM), do número da página que contem o endereço (PAG_MEM) e a verificação do tipo de página. Permitido o acesso, o "relógio" NUMERO_DE_ESTADOS será acrescido do valor t1 ou te, obtidos direta (caso de ROM e RAM) ou indiretamente (caso de EQU). Tentativas de escrita em ROM não serão levadas em conta (memória permanece inalterada), prosseguindo a simulação; um aviso de violação de memória será impresso na listagem final ao lado da instrução que a causou. O acesso a uma página inexistente com funcionamento ROM, produzirá sempre uma configuração de bits definida previamente pelo usuário.

fig 2.3



II.2.3 - Interrupções e Entradas/Saídas

O programador dispõe de três áreas dentro do programa principal onde pode introduzir instruções PL/1: no fim do módulo de declaração das variáveis, no módulo de interrupção e nas partes do módulo de entrada/saída que tratam do funcionamento do periférico.

Qualquer nova variável criada pelo programador, para utilização nas duas últimas áreas, pode ser declarada e inicializada no primeiro módulo do programa principal.

No módulo de interrupção pode-se estabelecer a ocorrência de interrupções e as instruções a elas associadas. Ao programador cabe definir:

- quantas interrupções diferentes existem
- a política de aparecimento de cada uma delas
- a instrução correspondente a cada interrupção

Esses três itens aparecem no módulo da seguinte maneira:

MODULO_INTERRUPT:

⋮

INTERRUPCAO_i: [cartões de definição da política]

IF política THEN DO;

[cartões de definição da política]

INTERRUPT = 1B;

VALOR_DA_INTERRUPCAO = i ;

```

GOTO MODULO_PCI@;

INT_PCI(i): REGI = 'xxxxxxxx'B;
[ NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+tpci; ]
GOTO MODULO_DECODIFICACAO;

[ INT_PCR_MM(i): REGB = 'bbbbbbbb'B;
  REGA = 'aaaaaaaa'B;
  [ NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+tpcrmm; ]
  GOTO MODULO_J_C@; ]
[ INT_PCR_I(i): REGB = 'bbbbbbbb'B;
  [ NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+tpcri; ]
  GOTO LABEL1; ]

END;

[ cartões de definição da política ]

⋮
GOTO MODULO_PCI@;

```

onde i é o número escolhido pelo programador para a interrupção.

As instruções PL/1 envolvidas por colchetes não são obrigatórias ou aparecem em casos particulares. Inicialmente algumas instruções PL/1 são usadas para o estabelecimento da política a ser testada e de eventuais condições ligadas a ela (é o grupo de cartões que será chamado CDPl). Um teste é feito sobre a política para saber se é verdadeira, ie, se haverá a interrupção. Em caso negativo, antes de examinar a próxima ($i + 1$), pode-se incluir mais alguns cartões que prepara-

rão a política para o futuro, sabendo que não ocorreu desta vez (é o grupo de cartões que será chamado de CDP3). Em caso afirmativo entra-se num DO privativo daquela interrupção. Novamente pode-se incluir cartões que prepararão a política para o futuro, sabendo que ocorreu a interrupção (é o grupo de cartões que será chamado de CDP2). A variável INTERRUPT passa a indicar que a 8008 está interrompida e VALOR_DA_INTERRUPCAO identifica a interrupção, necessário para a busca da instrução.

Tudo está pronto; o simulador, em vez de ir à memória buscar a primeira palavra, desviará para INT_PCI(VALOR_DA_INTERRUPCAO) onde se encontra este octeto (na listagem acima indicado como xxxxxxxx , e que deve ser definido pelo usuário). Eventualmente pode haver um retardo na busca da instrução e isto é simulado por um acréscimo na variável NUMERO_DE_ESTADOS. No caso de instruções longas, o simulador voltará ao módulo de interrupção em INT_PCR_I(VALOR_DA_INTERRUPCAO) para instruções de referencia imediata ou em INT_PCR_MM(VALOR_DA_INTERRUPCAO) no caso de instruções JMP e CAL. Para este último, REGB conterá a parte baixa do enderêço e REGA a parte alta. (na listagem acima indicados como bbbbbbbb e aaaaaaaa , e que devem ser definidos pelo usuário). Como em INT_PCI, retardos podem ser introduzidos em qualquer um dos dois.

No módulo de entrada/saída o programador dispõe de 32 áreas que correspondem a cada uma das 8 instruções de entrada e 24 de saída. Cada uma dessas áreas tem como primeiro cartão `MODULO_INP(i):` ou `MODULO_OUT(i):`, onde *i* é o número do periférico, e como último `GOTO MODULO_PCI;`. Nestas áreas pode-se introduzir qualquer sequência de instruções PL/1 que não violem a integridade do programa principal.

Para as áreas `MODULO_INP(i):`, o programador tem a seu dispor a variável `OUT_LATCH` que contém o valor do acumulador (o registro A da 8008). `OUT_LATCH` pode ser utilizado pelo periférico de entrada simulado, como um comando especial. Obrigatoriamente, a penúltima instrução deve ser a atribuição à variável `ACUMULADOR(1)` (o índice 1 é importante) de um determinado valor (é a simulação da leitura do dado pela 8008).

Para as áreas `MODULO_OUT(i):`, novamente tem-se `OUT_LATCH` como variável disponível ao programador. Conterá o dado que está sendo enviado ao periférico. Ao contrário de `MODULO_INP(i):`, é proibido a utilização da variável `ACUMULADOR(1)`.

Como no módulo de interrupção, pode-se simular os retardos envolvidos na chamada ao periférico. Esse tempo pode ser fixo ou calculado antes da instrução que altera a variável `NUMERO_DE_ESTADOS`.

No caso de periféricos que tem rotinas de simulação semelhantes, pode-se juntá-las numa área só e usar a variável VALOR_DO_PERIFERICO , que contem o número do periférico chamado, para diferenciar particularidades inerentes a cada um dos periféricos. Invariavelmente isto traz uma redução de cartões dentro do programa principal.

Em resumo, tem-se as áreas MODULO_INP(i): e
MODULO_OUT(i): programadas da seguinte forma:

```

      ⋮
MODULO_INP(j): [MODULO_INP(k):]
[cartões de definição do periférico]
[ NUMERO_DE_ESTADOS = NUMERO_DE_ESTADOS + tin;]
ACUMULADOR(1) = dado ;
GOTO MODULO_PCI;

      ⋮
MODULO_OUT(m): [MODULO_OUT(n):]
[cartões de definição do periférico]
[ NUMERO_DE_ESTADOS = NUMERO_DE_ESTADOS + tout;]
GOTO MODULO_PCI;

      ⋮
      , onde dado será 0 (00000000B) ou 255 (11111111B)
      (isto depende do circuito e da barra E) quando o periférico

```

inexistir. Nêste caso os cartões de definição do periférico não serão mais necessários.

II.3 - UTILIZAÇÃO DO SIMULADOR

Ao usuário é dada a possibilidade de definição do tipo e tempos de acesso das páginas do vetor memória. Isto é feito por meio de cartões de controle, colocados no início dos dados, através dos quais se estabelece a imagem da memória.

O primeiro cartão define o tamanho da página (cartão PAG) e os seguintes as características de cada página (cartões TIPO). Entretanto, a sua utilização é opcional, podendo o usuário deixar a cargo do programa principal a inicialização da imagem da memória. (o que os americanos chamam de "default").

Para limitar o tamanho da listagem final há o cartão de estabelecimento de pontos de quebra (cartão IMP), que indicará quais as instruções da memória que devem aparecer nos resultados.

O formato dos cartões de controle (PAG, TIPO e IMP) e dos dados está descrito no apêndice B.

O resultado da simulação consiste na impressão do estado interno da 8008 antes e após a resolução de cada instrução.

II.3.1 - Cartões de controle e programa

A primeira opção existente é a definição do tamanho da página, que deverá ser um número que é uma potência de dois e não poderá ultrapassar 16384 que é a capacidade máxima de endereçamento da 8008. Este valor deve ser igual ao tamanho da menor página existente na máquina real e será a unidade de divisão da memória. As páginas maiores serão subdivididas (e isto é possível já que o tamanho de todas as páginas é uma potência de dois) para se adaptar a essa unidade. Esta operação tem como resultado a divisão da memória em páginas de mesmo tamanho. Os cartões de controle para definição da imagem da memória, que seguem o cartão PAG, deverão se referir ao número das páginas, que variam de 0 (primeira página) até $16384/PAG-1$.

A não inclusão do cartão PAG implica na utilização pelo simulador de páginas de 256 palavras ($PAG = 256$) e, portanto, na existência de 64 páginas numeradas de 0 a 63.

Como outras opções, pode-se definir o tipo da página: ROM, RAM e inexistente. As inexistentes podem ter quatro comportamentos diferentes ao serem endereçadas:

- lê ou escreve numa página que existe
- só lê 11111111 (páginas tipo ROM/UM)
- só lê 00000000 (páginas tipo ROM/ZER)
- acesso proibido; para a simulação

O primeiro comportamento ocorre na prática quando se simplifica a decodificação do endereço (tipicamente em sistemas com pouca memória). Geralmente essa simplificação é feita desprezando-se os bits de mais alta ordem; assim 11000100110011 e 01000100110011 correspondem a uma mesma posição de memória.

Se não se simplificar a decodificação, chega-se aos dois comportamentos seguintes. A máquina tentará ler uma página que não existe e amostrará o que está na barra de entrada, neste momento em estado neutro, que pode ser 11111111 ou 00000000 dependendo dos circuitos utilizados.

O usuário só poderá definir os três primeiros casos ficando os não definidos com acesso proibido (isto só vale quando o usuário usa pelo menos um cartão TIPO).

Além da informação de tipo pode-se definir no cartão TIPO, tempos de acesso a páginas do tipo ROM, RAM, ROM/UM e ROM/ZER. Não se deve esquecer entretanto que esses tempos devem estar na unidade de tempo do simulador que é o período do sincronismo. Se uma memória tiver o tempo de escrita igual a 5,5 microsegundos (pior caso) e o sincronismo tem um período igual a 2,1 microsegundos, o `TEMPO_ESCRITA` para esta página será

$$\lceil 5,5 / 2,1 \rceil = 3 .$$

A não utilização de cartão TIPO implica na definição de uma memória de 16 K palavras do tipo RAM com acessos nulos e

dividida em 16384/PAG páginas.

Os programas do usuário para o simulador devem estar em binário e serão carregados em posições de memória determinadas pelo próprio usuário. Esses programas serão colocados após o cartão BRANCO.

O apêndice B mostra como organizar os cartões de dados (controle e programa) e descreve os formatos.

II.3.2 - Interpretação dos resultados da simulação

O resultado da simulação é uma série de linhas que correspondem à execução de instruções pela 8008. As linhas estão divididas em 18 campos, com informações sobre o processamento.

Da esquerda para direita tem-se:

- um campo para informação do nível do PC
- um campo para o conteúdo do PC naquele nível
- um campo para o nome da instrução que será executada (a primeira palavra da instrução está no endereço contido no campo anterior)
- quatro campos para as condições C, Z, S e P, após a execução da instrução
- sete campos para os registros A, B, C, D, E, H e L, já modificados pela instrução
- um campo para o valor da concatenação dos registros H e L (os dois bits mais significativos de H são desprezados)
- um campo para informações gerais: PULA, NAO PULA (para instruções JUMP, CALL e RETURN condicionais) e VIOL MEM (a violação de memória ocorre quando se tenta escrever em ROM)
- um campo usado pelas instruções de referência imediata e à memória; o campo mostra o valor lido
- um campo para o tempo "real" de processamento

A fim de evitar listagens muito compridas, pode-se limitar, através de pontos de quebra ("breakpoints"), uma região da memória, que conterá as instruções que poderão ser impressas. As instruções, cuja primeira palavra não estiver contida dentro dos limites estabelecidos pelos pontos de quebra, serão executadas normalmente, mas não aparecerão nos resultados (inexistência da linha).

A determinação dos pontos de quebra é feita pelo cartão IMP, que deve vir antes do cartão BRANCO. Até 7 pares de endereços (os limites mínimo e máximo) poderão ser definidos. A não inclusão do cartão IMP implica em que todas as instruções executadas pela 8008 aparecerão na listagem dos resultados.

As instruções sob interrupção são privilegiadas, aparecendo nos resultados independentemente dos pontos de quebra. Diferencia-se as instruções executadas normalmente daquelas sob interrupção, pelo fato que nas últimas não aparece o valor do nível e do conteúdo do PC, ficando os dois campos em branco.

Maiores detalhes sobre o cartão IMP encontram-se no apêndice B.

II.4 - CONCLUSÃO

O programa principal é modular no sentido que os módulos tem funções bem definidas dentro do programa principal. No caso dos módulos de simulação da 8008 a modularidade significa também que a posição entre eles é irrelevante.

O número máximo de interrupções que é possível definir dentro do programa principal é dado pelo tamanho dos vetores "label" INT_PCI, INT_PCR_I e INT_PCR_MM. É claro então, que para aumentar o número de interrupções basta alterar o cartão de declaração dessas três variáveis.

Uma crítica que pode ser feita à programação do módulo de interrupção é uma inerente hierarquia entre as interrupções, mais especificamente, as primeiras examinadas, serão tratadas imediatamente, em detrimento das seguintes, mesmo que essas últimas tenham ocorrido antes. Ora, há casos na prática onde isto não ocorre, onde, por exemplo, a interrupção que chega primeiro é a primeira a ser tratada, independentemente de prioridade. Isto pode ser simulado no módulo de interrupção, substituindo-se o cartão `GOTO MODULO_PCI@;`, que segue o cartão `VALOR_DA_INTERRUPCAO = i ;`, por um `GOTO INTERRUPCAO(j);` (se as interrupções forem colocadas em ordem numérica crescente, então $j = i + 1$). Os CDP2 da interrupção j tratarão de comparar os tempos de ocorrência dela e da anterior. Desta maneira pode-se varer todo o módulo de interrupção antes de determinar

qual o VALOR_DA_INTERRUPCAO .

Será preciso introduzir dentro do programa principal, alguma condição que termine a simulação. Isto pode ser feito pelo módulo de interrupção, bastando contar o número de instruções HALT que passam pela 8008 (uso dos CDP1). Atingido um certo número a simulação é terminada. Outra maneira é utilizar uma instrução de OUT, que quando chamada envie o programa para seu fim (GOTO ENDE;). Em casos onde o risco de "loops" intermináveis é grande, é de interesse limitar a simulação por meio do relógio interno, ie, testa-se se NUMERO_DE_ESTADOS ultrapassou um dado valor. O mais seguro parece ser combinar esta última maneira com uma das duas primeiras.

Sempre com a preocupação de adaptar o simulador à realidade, as duas instruções inexistentes, cujo funcionamento na máquina foi determinado experimentalmente, foram introduzidas no simulador com os seguintes nomes:

- SET101 (00 111 000) altera somente os "flip-flop" de condição Z, S e P, forçando-os para os níveis 1, 0 e 1 , respectivamente
- SET010 (00 111 001) altera somente os "flip-flop" de condição Z, S e P, forçando-os para os níveis 0, 1 e 0 , respectivamente

O tempo para as duas é 5 unidades de tempo do simulador.

Infelizmente o simulador não oferece qualquer facili-

dade para reconhecimento de mnemônicos e montagem de programa. Essas deficiências poderiam ser cobertas por futuros trabalhos que envolveriam a programação de um montador ou mesmo de um compilador para uma linguagem de alto nível.

O programador não deve esquecer que inicializado o simulador não se sabe o conteúdo certo dos "flip-flop" de condição. O mesmo acontece com a memória antes do carregamento dos programas e constantes (o simulador, como a máquina real, é capaz de interpretar e executar configurações binárias presentes em posições da memória não inicializadas pelo programa). Só se garante como conteúdo o valor 0, toda a pilha de PC e os registros A, B, C, D, E, H e L. O programador deve atentar para essas particularidades para não incorrer em erros de programação.

C A P Í T U L O I I I

A REALIZAÇÃO

III.1 - INTRODUÇÃO

O projeto, inicialmente um microcomputador de 8 bits com capacidade de 16K palavras de memória, 8 periféricos de entrada, 24 de saída e um painel, teve seu tamanho reduzido, por motivos de disponibilidade de material e de tempo, a uma máquina de 8 bits com configuração máxima de 256 palavras de memória, 1 periférico simples de entrada, 2 de saída e o painel, projetado em função dessas simplificações. Somente o processador, apresentado no primeiro capítulo, foi montado na sua forma geral, pois o teste de sua lógica era um dos objetivos mais importantes do trabalho. As limitações tiveram efeito sobre o projeto da memória, das entradas e saídas, e do painel, que no caso se confundiu com o sistema de interrupção.

O projeto tinha e teve como meta a criação de um painel acoplado ao processador que permitisse ao usuário, o acesso à memória e ao estado interno da 8008 (por estado interno entende-se a pilha de contadores de programa, os registros de indexação e os quatro "flip-flop" de condição). Este painel é constituído por lâmpadas para a visualização das informações, teclas (chaves do tipo reversível) para comando do microcomputador e chaves (tipo interruptor) para dados em geral.

Nas seções seguintes explicar-se-á o projeto do computador e depois as simplificações introduzidas no protótipo. Os circuitos montados estão no apêndice A .

III.2 - PROJETO DO MICROCOMPUTADOR

O microcomputador projetado tem por base a arquitetura descrita no capítulo I. Para o primeiro protótipo escolheu-se uma configuração bem simples: apenas os periféricos de saída 17 e 37, e o de entrada 7. Os periféricos de saída são simples registros, sendo que um deles (OUT 17) servirá também como periférico de entrada (INP 7). O outro (OUT 37) estará ligado a lâmpadas no painel.

O painel é dentro do sistema o único elemento capaz de interromper a 8008. Estará encarregado da geração das diversas interrupções e instruções correspondentes. É por meio dele que o operador comandará o funcionamento da máquina.

III.2.1 - O painel

Inicialmente, é preciso notar a impossibilidade do acesso direto aos registros internos da 8008. O projetista só dispõe da barra de dados por onde flui tôdas as informações, dos sinais de controle e do repertório de instruções. Para obter o estado interno deve-se então recorrer a métodos indiretos.

No caso dos registros de indexação (A, B, C, D, E, H e L) pode-se exclusivamente por programação (a maneira mais prática), utilizando as instruções sôbre registros e de entrada/saída, mover um a um seus conteúdos para registros periféricos. Esta solução não é aplicável para os contadores de programa, ie, a pilha de PC (program counter), já que as instruções que agem sôbre êles (JUMP, CALL, RESTART e RETURN) somente o fazem alterando os conteúdos. Porém sabe-se que o valor do contador de programa é enviado para fora sempre que a 8008 busca uma instrução. Basta portanto, amostrar o valor de RL e RH quando contiverem o enderêço desejado, independentemente da instrução processada; é, ao contrário dos registros de indexação, uma solução puramente de circuito. Diferentemente, os "flip-flop" de condição exigem uma solução mista: estarão presentes na barra de dados para serem amostrados, somente durante o estado T4 do ciclo PCC de qualquer instrução INP. O registro de 4 bits onde serão armazenados C, Z, S e P estará ligado a lâmpadas.

Além das quatro condições, é importante visualizar

também o conteúdo da pilha de PC e os registros de indexação, porém seria dispendioso montar oito conjuntos de lâmpadas para cada elemento da pilha e mais sete para cada um dos registros. Limitou-se o painel a dois conjuntos apenas que mostrarão o valor do PC usado naquele instante e um dos sete registros, selecionado por uma trinca de chaves.

Para completar o órgão de visualização, duas lâmpadas informarão ao operador do microcomputador, quando a máquina está parada (STOP) e em estado de espera (WAIT).

A configuração final consta então de quatro grupos de lâmpadas:

- uma linha de 14 para o conteúdo do PC
- uma linha de 8 para um registro de indexação
- uma linha de 4 para as condições C, Z, S e P
- duas lâmpadas para WAIT e STOP

Por questão de economia, os dois primeiros grupos serão usados pela parte do painel que faz o acesso à memória (considere que a configuração acima tem um custo equivalente a aproximadamente 40% do preço da CPU 8008). A linha de 14 mostrará agora o endereço da memória e a de 8 o dado lido ou escrito neste endereço.

Como o painel não tem ligação direta com a memória, a leitura e escrita será feita utilizando a 8008 que se encarre-

gará de ler as informações do painel (dado e endereço) e devolver ao mesmo o octeto lido.

É imprescindível que tanto êsses acessos à memória por meio da 8008 como a obtenção do estado interno não modifique êste estado. O operador deve ser capaz de parar a máquina durante um programa, ler a memória, observar o estado interno e dar a ordem de continuação de processamento, sem que nenhum bit da 8008 seja alterado. Ora, como o processador participa ativamente nas operações de painel mencionadas, a obediência a esta condição de inviolabilidade da 8008 torna-se a parte mais crítica do projeto. Isto implica de imediato que a comunicação painel-processador deve ser feita sob interrupção; única situação em que a pilha de PC não tem seu conteúdo modificado por instruções outras que JUMP, CALL, RESTART e RETURN.

Tudo depende então de se associar a cada tecla do painel uma interrupção, ou uma sequencia ininterrupta de interrupções, (o que significa dizer que a 8008 permanecerá, durante tôda a sequencia, gerando T1I em vez de T1); ao sinal de interrupção inicial seguir-se-á, antes do fim do processamento da instrução correspondente a essa primeira interrupção, um novo sinal de interrupção, com sua instrução, e assim até o fim da sequencia. Durante a execução da última instrução da sequencia não haverá geração de interrupção.

Do funcionamento da 8008 sabe-se que a cada interrup-

ção, ou sequência de interrupções, estará associada uma instrução, ou sequência de instruções. Surge então o problema de geração dessas instruções, pois que não estarão na memória. Escolheu-se como maneira de produzi-las a matriz programável (MAPRO). Essa matriz funciona como uma pequena memória de leitura exclusiva e tem seu próprio contador de endereço. É da MAPRO que a 8008 lerá as instruções correspondentes às interrupções geradas pelo painel. Cada interrupção terá sua instrução armazenada na MAPRO em posição bem definida. Acionar uma tecla de comando do painel significará carregar no contador de endereço da MAPRO um valor que aponta para a instrução relativa à interrupção gerada por esta tecla. No caso de uma sequência de interrupções, o contador será incrementado a medida que as palavras, colocadas sequencialmente na MAPRO, forem lidas, gerando a sequência de instruções anteriormente mencionada .

Esses programas de painel, se assim se pode chamá-los, contidos na matriz programável, serão executados com a 8008 interrompida, cabendo à própria MAPRO a geração dos sucessivos sinais em L.I. (a tecla só é responsável pela primeira interrupção que dá partida ao processo).

III.2.2 - Teclas

Dividir-se-á as teclas em dois grupos: para memória e para processador. As primeiras, quando acionadas, farão com que as lâmpadas do painel funcionem em Modo Memória, ie, o conjunto de 14 lâmpadas mostrará o endereço onde se quer ter acesso e o de 8 o dado lido:

- tecla END (endereço) - colocação do endereço, indicado por chaves, num registro de 14 bits denominado registro de painel (RP)
- tecla ESC (escrever) - escrita na posição de memória, cujo endereço é dado por RP, do dado que está afixado nas chaves (as 8 inferiores do grupo de 14 usadas para o endereçamento) (chaves imediato)
- tecla +1 (incremento) - $RP \leftarrow RP + 1$

A não existência de uma tecla LER decorre da leitura automática da memória provocada por qualquer uma das 3 teclas acima.

As teclas para processador farão com que as lâmpadas funcionem no Modo Cpu, ie, o conjunto de 14 mostrará o PC e o de 8 algum registro:

- tecla EST (estado) - obtenção do estado interno da 8008. As chaves para selecionamento do registro obedecem a convenção da 8008 (A-000, B-001, ...) (chaves registro)

- tecla PAS (passo) - resolução da próxima instrução do programa em memória. Durante a execução desta instrução a 8008 não estará interrompida.
- tecla IP (instrução painel) - execução sob interrupção da instrução afixada por 8 chaves (chaves instrução). Esse octeto será concatenado com as 14 chaves já existentes para formação da instrução longa.

Os programas de painel para essas teclas sempre terminarão por uma instrução HALT. Além das seis acima, mais duas, PARE e CONT (continue), se relacionam apenas com o funcionamento do processador.

III.2.3 - Matriz programável

A matriz programável do painel (MAPRO) é uma memória de leitura exclusiva a diodos. Possui um decodificador que selecionará uma das linhas da matriz. O endereçamento é feito por meio de um contador que pode ser carregado com qualquer valor.

A MAPRO terá seu contador incrementado cada vez que a 8008 pedir uma palavra de instrução do painel (figura 3.1). Entre T11 e T3A há tempo suficiente para completar a leitura da palavra no novo endereço. O nível painel serve para diferenciar o painel de outras partes do sistema de interrupção.

A matriz tem nove colunas. Oito destinadas à instrução e uma para a interrupção. Nesta última, a colocação de um diodo significará que não há sinal de interrupção no fim da instrução. No caso de instruções longas, que ocupam mais de uma linha na MAPRO, basta colocar o diodo na última palavra da instrução para inibir a interrupção.

As teclas do painel carregarão no contador o endereço do programa de painel correspondente menos um (a razão é que reconhecida a interrupção e chamado o painel, o contador da MAPRO será incrementado, sobre o valor carregado pela tecla, antes da leitura da instrução) e gerarão um sinal de interrupção inicial.

Os programas na MAPRO terminarão normalmente por uma

instrução de HALT, que não deve gerar interrupção (na MAPRO corresponde a uma linha com apenas um diodo na coluna de microprogramação da interrupção).

Os programas poderão ter partes comuns, mas pulos dentro da matriz devem ser realizados por meio de circuitos junto ao contador.

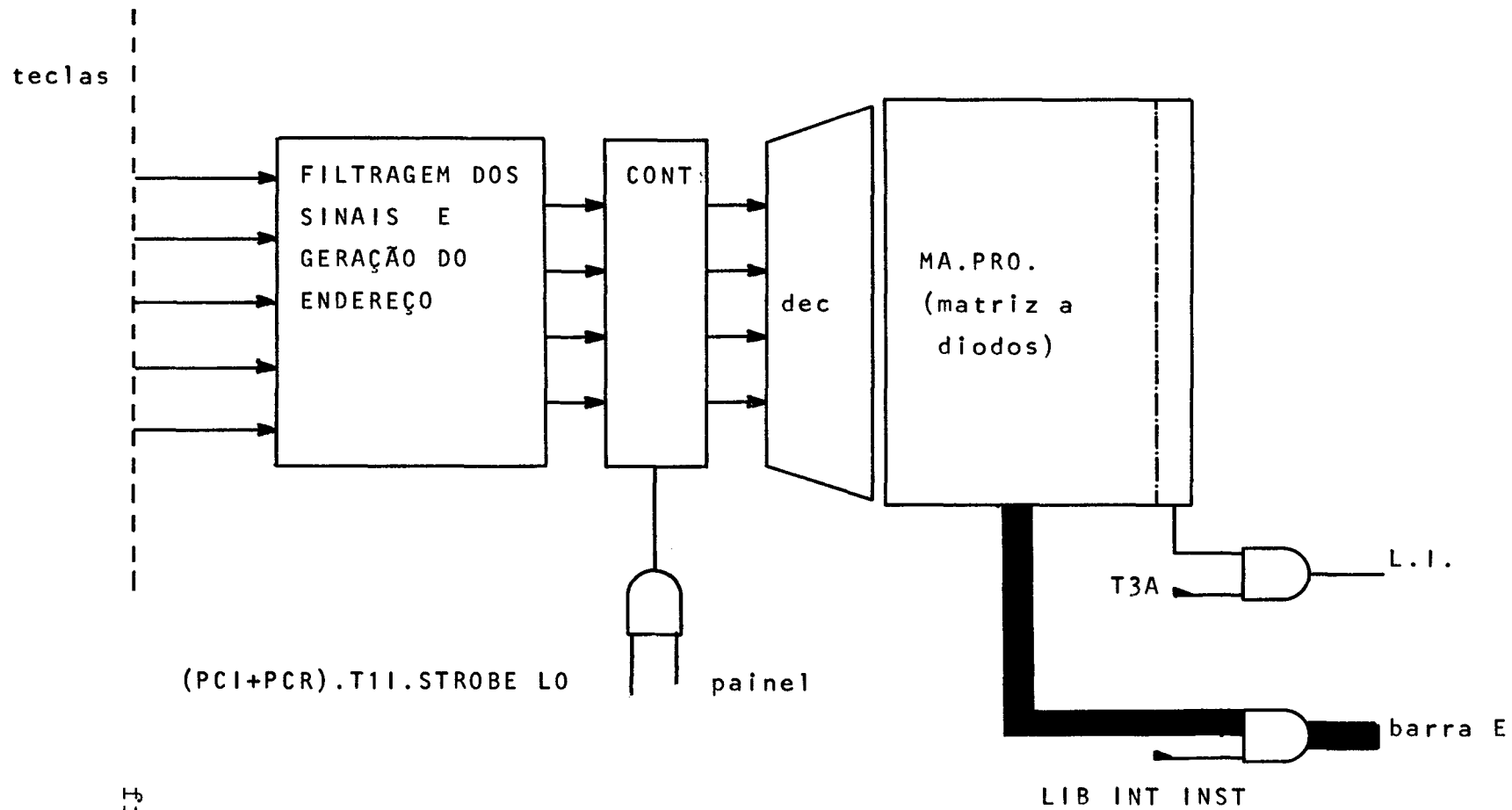


fig 3.1

III.3 - IMPLEMENTAÇÃO

A implementação do protótipo se diferencia do projeto inicial pela redução do enderêço de 14 para 8 bits, o que influencia toda a fiação, pelas simplificações que introduz nos circuitos combinacionais, e especialmente na organização do painel.

A matriz programável que armazenaria todas as palavras das instruções dos programas de painel, agora passa a fornecer somente o primeiro octeto, ficando a geração do segundo a cargo de um outro circuito (isso se deve à limitação do espaço para colocação dos diodos na placa onde foi montada a MAPRO). É claro, que, como o enderêço é só de 8 bits, a eventual terceira palavra é desnecessária ("don't care"); pela mesma razão, o registro H da 8008 perde sua importância.

Os circuitos de decodificação do código de periférico foram minimizados aproveitando a existência de poucos periféricos.

A memória sendo de apenas 256 palavras, a decodificação do enderêço ficou a cargo do próprio módulo de memória (o protótipo só tem um módulo de 256 palavras). Também, o enderêço estando pronto em T1, ie, bem antes da leitura do octeto da memória pela 8008 em T3A, não houve necessidade de se trabalhar com o sinal RDY, que permanecerá sempre ativo.

Quanto ao painel, a largura de RP foi reduzida para oito, evitando a multiplexação das partes alta e baixa sôbre a barra E que ocorre quando se usa o RP de 14 bits.

III.3.1 - Organização do protótipo

A organização do protótipo é o da figura 3.2 (é interessante compará-la com a figura 1.7). Os periféricos de saída ficaram restritos a dois registros de 8 bits: um ligado a lâmpadas que será usado tanto para mostrar o valor de um registro (Modo Cpu), como o valor lido na memória (Modo Mem), e o outro servindo também como periférico de entrada. Essa última ligação permitirá salvar o conteúdo do acumulador durante os programas de painel que o utilizam para transferências. O sistema de interrupção é formado pela MAPRO, que contém os programas de painel e que é inicializada através das teclas (carregamento do endereço inicial no contador da MAPRO, que faz parte do controle da MAPRO) (figura 3.3). As chaves instrução usadas pela tecla IP, e a trinca de chaves para seleção do registro, fazem parte integrante da matriz de diodos, por estarem programando diretamente (colocação ou não de diodos nas interseções de linhas com colunas) linhas da matriz. O segundo octeto das instruções longas é obtido, ou do registro de painel (RP), ou diretamente das chaves imediato (esse nome não

significa que elas sejam usadas exclusivamente por instruções de referencia imediata; instruções sôbre o PC também usarão as mesmas chaves para o enderêço). A matriz se encarregará de seleccionar qual dos dois octetos deve ser colocado na barra E (LIB INT IMEDIATO e LIB INT ENDEREÇO).

O registro RP, que está ligado às chaves imediato, mostrará em Modo Mem o enderêço que foi carregado pela tecla END (END gera SET IMEDIATO) ou incrementado pela tecla +1. ESC não atua sôbre RP. No Modo Cpu, RP mostrará o valor do PC carregado por SET PC (gerado pelo processador sob ordem da matriz de diodos).

Os circuitos em detalhe se encontram no apêndice A.

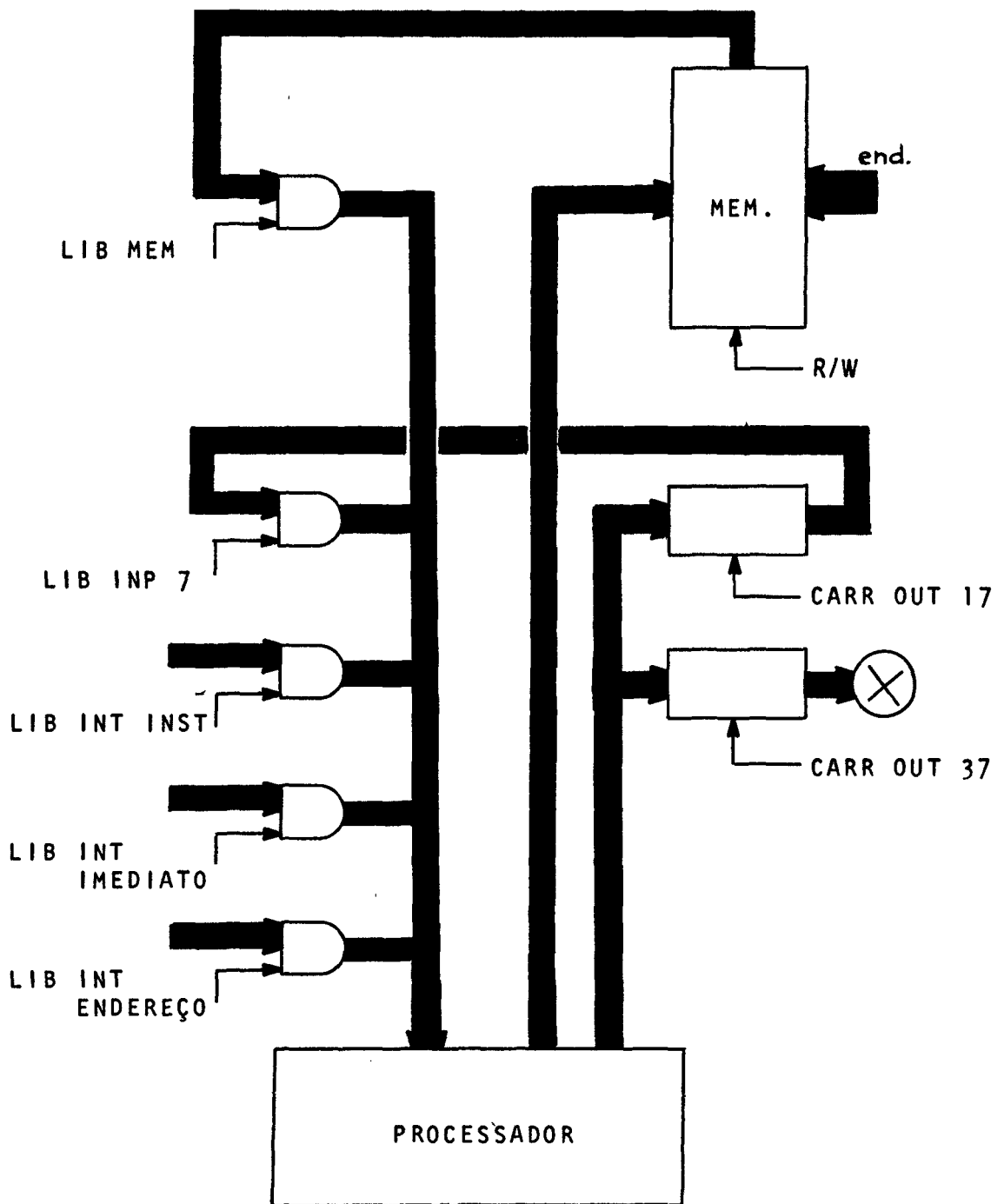


fig. 3.2

PAINEL

chaves instrução
chaves registro

teclas controle

chaves imediato

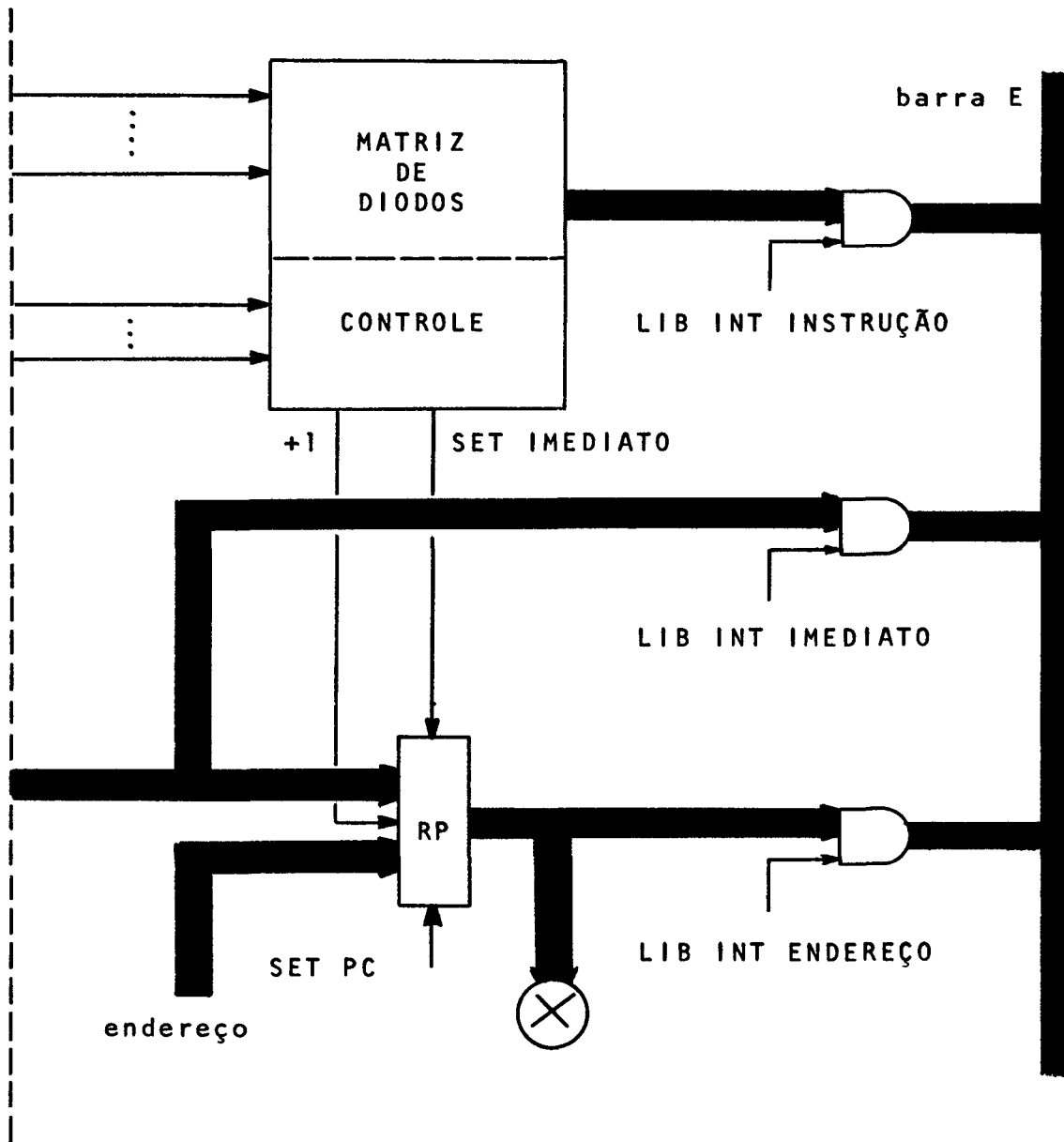


fig 3.3

III.3.2 - Programas de painel

São os seguintes os programas de painel das teclas do microcomputador (o asterisco a frente da instrução significa que é gerada uma interrupção ao final da execução):

- tecla EST (estado)

*OUT_17 o acumulador é salvaguardado no periférico 17; durante a execução desta instrução, o conteúdo do PC é amostrado (SET PC) em RP

*LAr transferência do registro r , selecionado pelas chaves registro, para o acumulador

*OUT_37 colocação do conteúdo do registro, agora também presente no acumulador, no periférico 37

*INP_7 recuperação do acumulador

HLT fim do programa; 8008 em STOP

- tecla IP (instrução painel)

*(instrução do painel) é a instrução programada nas chaves instrução (primeiro octeto) e nas chaves imediato (octeto da instrução longa)

*OUT_17

*LAr

*OUT_37

*INP_7

HLT

- tecla PAS (passo)

LLL nada faz; serve apenas para que a próxima
instrução não seja executada sob interrup-
ção

*(próxima instrução na memória) é a execução de mais
uma instrução do programa em memória (só pos-
sível com a 8008 não interrompida)

*OUT_17

*LAr

*OUT_37

*INP_7

HLT

- tecla PARE

HLT para o processamento

- tecla CONT (continue)

LLL instrução fantasma que tira a 8008 do esta-
do de STOP

- tecla END (enderêço)

inicialmente RP é carregado (SET IMEDIATO) com o va-
lor das chaves imediato

*OUT_17 salvaguarda o acumulador

*LHL salvaguarda o registro L em H

*LLI(RP) carrega em L o valor contido em RP (a ati-

vação desta instrução na matriz programável colocará RP na barra E quando a 8008 solicitar o segundo octeto da instrução)

*OUT_37 instrução fantasma; necessária para compatibilizar este programa com o da tecla ESC

*LAM leitura do dado

*LLH recuperação do registro L

*OUT_37 colocação do dado no periférico 37

*INP_7 recuperação do acumulador

HLT

- tecla +1

inicialmente RP é incrementado

*OUT_17

*LHL

*LLI(RP)

*OUT_37

*LAM

*LLH

*OUT_37

*INP_7

HLT

- tecla ESC (escrever)

gera o sinal ESC até o fim do programa

*OUT_17

*LHL

*LLI(RP)

*LMI(chaves imediato) essa instrução substitui a

OUT_37 nos programas de painel das teclas

END e +1, quando o sinal ESC está no nível

um lógico

*LAM

*LLH

*OUT_37

*INP_7

HLT

A organização dos programas na matriz programável é a seguinte:

teclas END, +1 e ESC ----->	0000	*OUT_17
	0001	*LHL
tecla PAS -->	1000	LLL
		0010 *LLI(RP)
	{ 1001 *(inst.mem.)	{ 0011 *OUT_37
tecla IP ---->	{ 1001 *(inst.painel)	{ 0011 *LMI(imed.)
tecla EST -->	1010	*OUT_17
	0100	*LAM
	1011	*LAr
	0101	*LLH
	1100	*OUT_37
	1101	*INP_7
tecla PARE ->	1110	HLT
tecla CONT ->	1111	LLL

O contador de endereço da MAPRO é programado por circuito para passar de 0101 a 1100 automaticamente. A instrução em 0011 é escolhida pelo sinal ESC. Em 1001 a diferença é a existência ou não da interrupção.

Para facilitar o projeto da MAPRO, no protótipo, preferiu-se incrementar o contador de endereço logo após a leitura da instrução. O apêndice A indica qual o sinal de avanço do contador para este caso e a posição dos diodos de programação da interrupção

Como a primeira instrução do programa é sempre processada sob interrupção, foi necessário introduzir, no programa da tecla PAS, uma instrução LLL ("no operation") que não gera interrupção.

III.4 - CONCLUSÃO

Necessitando de poucas instruções, portanto de poucas palavras, o painel encontra na matriz programável uma solução econômica e flexível, pois possibilita a inclusão, alteração ou substituição de funções de comando (são as teclas no painel), com relativa facilidade. Poder-se-ia chamá-lo de painel micro-programado, pois a coluna da interrupção na matriz tem uma função de controle para o painel.

Os programas das teclas END, +1 e ESC beneficiaram das simplificações introduzidas na implementação do microcomputador (os programas de painel para o Modo Cpu estão implementados de maneira geral). A sua generalidade foi violada quando se salvaguardou o registro L no H. Para o projeto ideal deve-se salvarguardar H e L em registros periféricos como foi feito com o acumulador. Há diversas maneiras de fazê-lo. A mais direta é utilizar novas instruções de entrada e saída para ler e escrever o conteúdo de H e L num registro periférico:

```
*OUT_17  salvaguarda A
```

```
*LAL
```

```
*OUT_16  salvaguarda L
```

```
*LAH
```

```
*OUT_15
```

```
⋮
```

```

:
:
*INP_5
*LHA      recupera H
*INP_6
*LLA      recupera L
*INP_7    recupera A

```

O problema com a solução acima é a utilização de três dos oito periféricos de entrada disponíveis. Com alguns circuitos combinacionais consegue-se ler H, L e A pela mesma instrução de entrada, bastando utilizar o código presente no acumulador para diferenciar entre os três registros:

```

*OUT_17
*LAL
*OUT_16
*LAH
*OUT_15
:
:
:LAI 5    cinco é o código escolhido para o registro H
*INP_7
*LHA
*LAI 6    seis é o código escolhido para o registro L
*INP_7
*LLA

```

*LAI 0 zero é o código escolhido para o acumulador
 *INP_7

Outras soluções existem, que utilizam a técnica de amostragem (como foi usada para o PC em RP), mas só devem ser aplicadas quando houver limitação na capacidade de memória da MAPRO.

Quanto aos circuitos montados no protótipo, e que estão detalhados no apêndice A, deve-se observar os seguintes fatos:

- o processador foi montado de acordo com o projeto do primeiro capítulo, apenas sendo usados o RL para o endereçamento da memória e alguns bits de RH para o código de periférico.
- no controle da multiplexação da barra E, introduziu-se um circuito sequencial que separa as instruções de referencia imediata daquelas de referencia a memória, permitindo assim a execução correta das instruções LrM sob interrupção. Esse circuito foi incorporado ao projeto para dar generalidade a MAPRO. É bastante simples pois tira partido do fato que a diferença entre as instruções de referencia memória e referencia imediata pode ser detetada apenas pelo bit D0 do primeiro octeto da instrução.
- na MAPRO a diferença mais importante com o modelo ideal, é o avanço do contador de endereço ser feito logo após a leitura da instrução. Isto implica

em que no protótipo o endereço carregado pela tecla é o da primeira instrução, e não o da anterior a essa. Aparentemente uma melhor solução, o avanço adotado na implementação traz problemas na geração da interrupção, pois no fim da execução de uma das instruções do painel (que geralmente vem acompanhado por uma geração de interrupção para que a próxima também seja processada sob interrupção), quem deve gerar o sinal em L.I. não é mais a linha dessa instrução que termina, mas a próxima linha, já que o contador aponta para ela. É certamente uma má solução para o caso geral da MAPRO, mas no protótipo, a custa de uma porta lógica para resolver uma das incongruências deste avanço no caso da geração de interrupção, permitiu uma importante simplificação dos circuitos de geração de endereço e do contador.

Os testes efetuados programando-se o microcomputador pelo painel permitiram a obtenção dos seguintes importantes resultados:

- as instruções de rotação do acumulador estão incompletamente especificadas no manual da INTEL. Em vez de

00 000 010 RLC

00 001 010 RRC

00 010 010 RAL

00 011 010 RAR

o correto é

00 X00 010 RLC

00 X01 010 RRC

00 X10 010 RAL

00 X11 010 RAR

onde X pode ser 0 ou 1.

- as duas instruções inexistentes 00 111 000 e 00 111 001 , que logicamente deveriam corresponder a mnemônicos INM e DCM, ie, incrementação ou decrementação do dado em memória, não atuam sobre a memória, pois pertencem ao grupo de instruções que só tem o ciclo PCI. Com um ciclo só não é possível se ler e reescrever um dado na memória. Entretanto, a 8008 interpreta esses octetos como instruções de incrementação e decrementação; mas, sobre que valor, já que não se tem acesso a memória e nenhum registro foi especificado? O que acontece, é que a ALU (arithmetic and logic unit) vai operar sobre o que está na barra interna a 8008. Como a barra estará neutra (11111111), a incrementação produz 00000000 e a decrementação 11111110, valores que são perdidos pois não são carregados em nenhuma parte. Mas, o cálculo desses valores na ALU provoca a alteração dos "flip-flop" de condição de acordo com o resultado. Na incrementação, como o resultado é zero, Z=1, S=0 e P=1

(as instruções de incrementação e decrementação não alteram a condição C). Para a decrementação, como o resultado é 254, Z=0, S=1 e P=0. Foram escolhidos dois mnemônicos para essas instruções: SETl0l e SET0l0 (INM e DCM não foram adotados para não confundir o programador desavisado). SETl0l e SET0l0 são instruções de um ciclo só (PCI) constituído pelos estados Tl (ou TlI), T2, T3, T4 e T5; exatamente como INr e DCr.

O protótipo é um ponto de partida para outros trabalhos que serão feitos usando a 8008. As atividades que devem ser levadas adiante agora são:

- projeto e implementação de um programador de PROM (Programmable Read Only Memory). Este projeto deve ser suficientemente flexível para permitir a programação da maioria das PROM comerciais.
- montagem definitiva do processador em placas de circuito impresso, com fontes própria de +5 Volts e -9 Volts.

Em seguida devem ser realizados a ligação do microcomputador a uma TTY, um CRT e um K-7 (fita magnética). Já se encontram em fase de estudo a ligação de uma máquina de escrever, de um dispositivo de visualização e de uma fita "cassete".

C O N C L U S ã O

No primeiro capítulo descreveu-se uma organização de barras para um processador que usa a INTEL 8008. Os circuitos propostos foram implementados e testados com resultados inteiramente satisfatórios.

O simulador do segundo capítulo foi testado um grande número de vezes nos computadores IBM/360 e IBM/370 do Núcleo de Computação Eletrônica e demonstrou ser um instrumento útil para o programador da 8008. Uma versão simplificada do simulador pode ser feita retirando-se do programa principal as facilidades oferecidas pelos cartões de controle.

No terceiro capítulo apresentou-se os problemas práticos ocorridos durante a realização. Fatores econômicos, que pesam muito numa engenharia cara como a de computadores, forçaram compromissos entre o ideal e o real. Mesmo assim procurou-se utilizar material de boa qualidade (especialmente no que concerne o "hardware", como chaves, teclas, etc.), a fim de evitar eventuais defeitos intermitentes, que são o pesadelo do projetista.

A utilização de placas para fiação "wire-wrap", aliada a um trabalho de bancada sistemático, com documentação detalhada e atualizada dos circuitos e fiação, e esquemas de teste cuidadosamente elaborados, permitiu que a montagem e testes finais

fossem concluídos após quatro meses do seu início.

A montagem do microcomputador foi de grande utilidade para o melhor conhecimento da 8008 (só conhecemos realmente um computador quando sentamos frente a êle). Com o protótipo pode-se corrigir e completar o material informativo do manual da INTEL, cuja primeira edição contém inúmeros êrros, e adaptar o simulador a todas as idiossincrasias da 8008, tornando-o mais completo que o próprio manual.

O universo dos microprocessadores não se restringe à 8008. Outras companhias já lançaram modelos mais ou menos sofisticados que a CPU da INTEL (a própria INTEL já se superou com o microprocessador 8080), e a tendencia é que mais e mais empresas entrem nêste mercado. A grande incógnita hoje é o futuro dos micros frente aos minis, ou, quem sabe, o futuro dos minis, que vêem seu império atacado por anões. Talvez se encontre a resposta no fato que várias companhias de minicomputadores já incorporaram circuitos LSI de grande complexidade a suas máquinas (será que passarão a se chamar microcomputadores?).

O certo de tudo isso é que os microprocessadores abriram uma nova frente no desenvolvimento de sistemas digitais, e o projetista atualizado sempre os incluirá na lista de possíveis soluções.

B I B L I O G R A F I A

1. "8008 8-bit Parallel Central Processor Unit" Intel Corp.,
November 1972.
2. Bruce Gladstone "Designing with microprocessors instead of
wired logic asks more of designers", Electronics, October 11,
1973, p.91.

A P Ê N D I C E A

CIRCUITOS

Ilustração dos circuitos do protótipo.

figura A.1 Processador

Unidade central de processamento, "interface" TTL-MOS
MOS-TTL para a INTEL 8008 e decodificação do estado
("status bits" S0, S1 e S2).

figura A.2 Processador

Obtenção dos ciclos, sinais de carregamento para RL e
RH e amostragem da barra E.

figura A.3 Processador

Registros RL e RH

figura A.4 Processador

Geração do pulso de escrita em memória e sincronização
do sinal de interrupção.

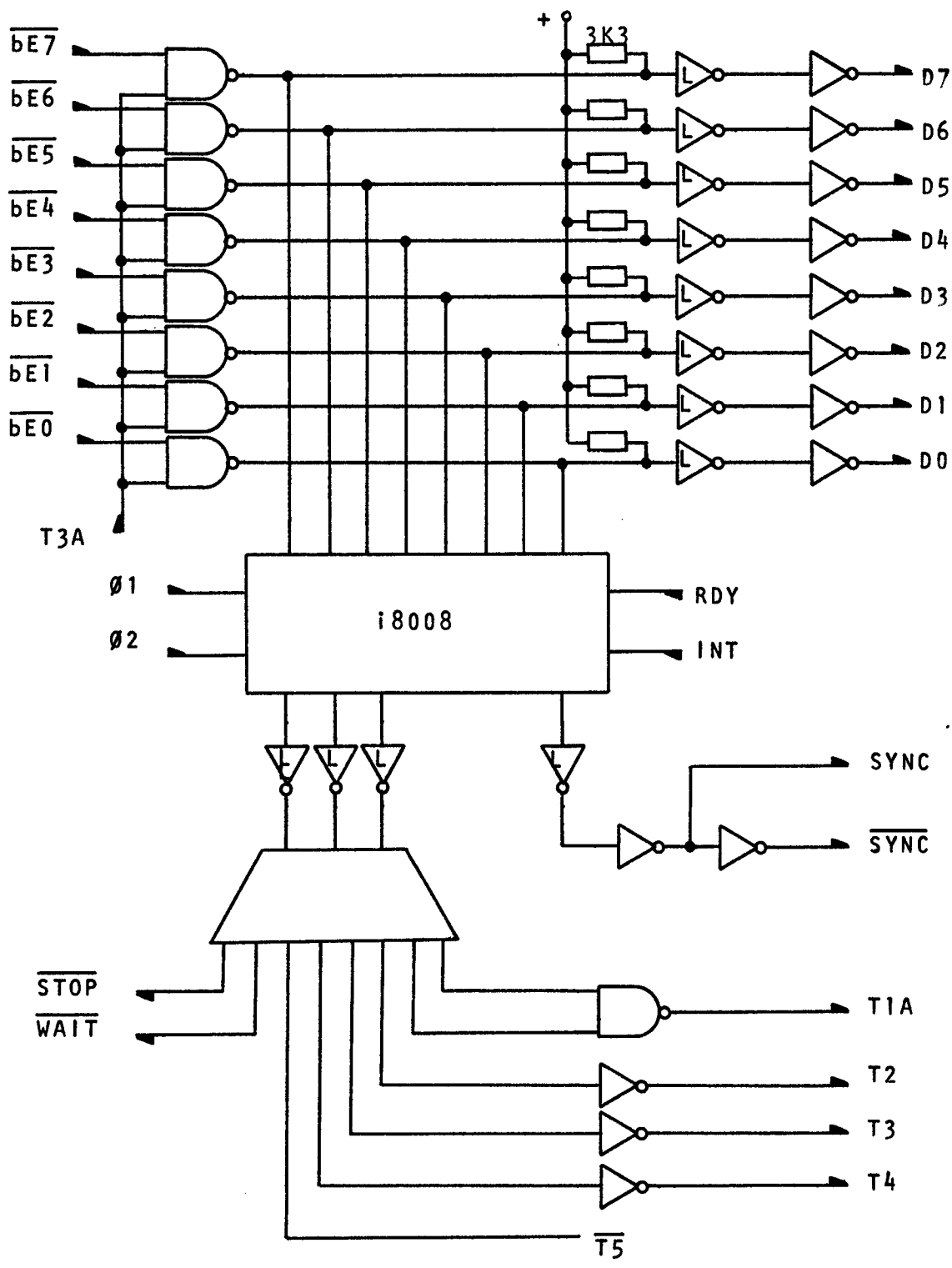


fig.A.1

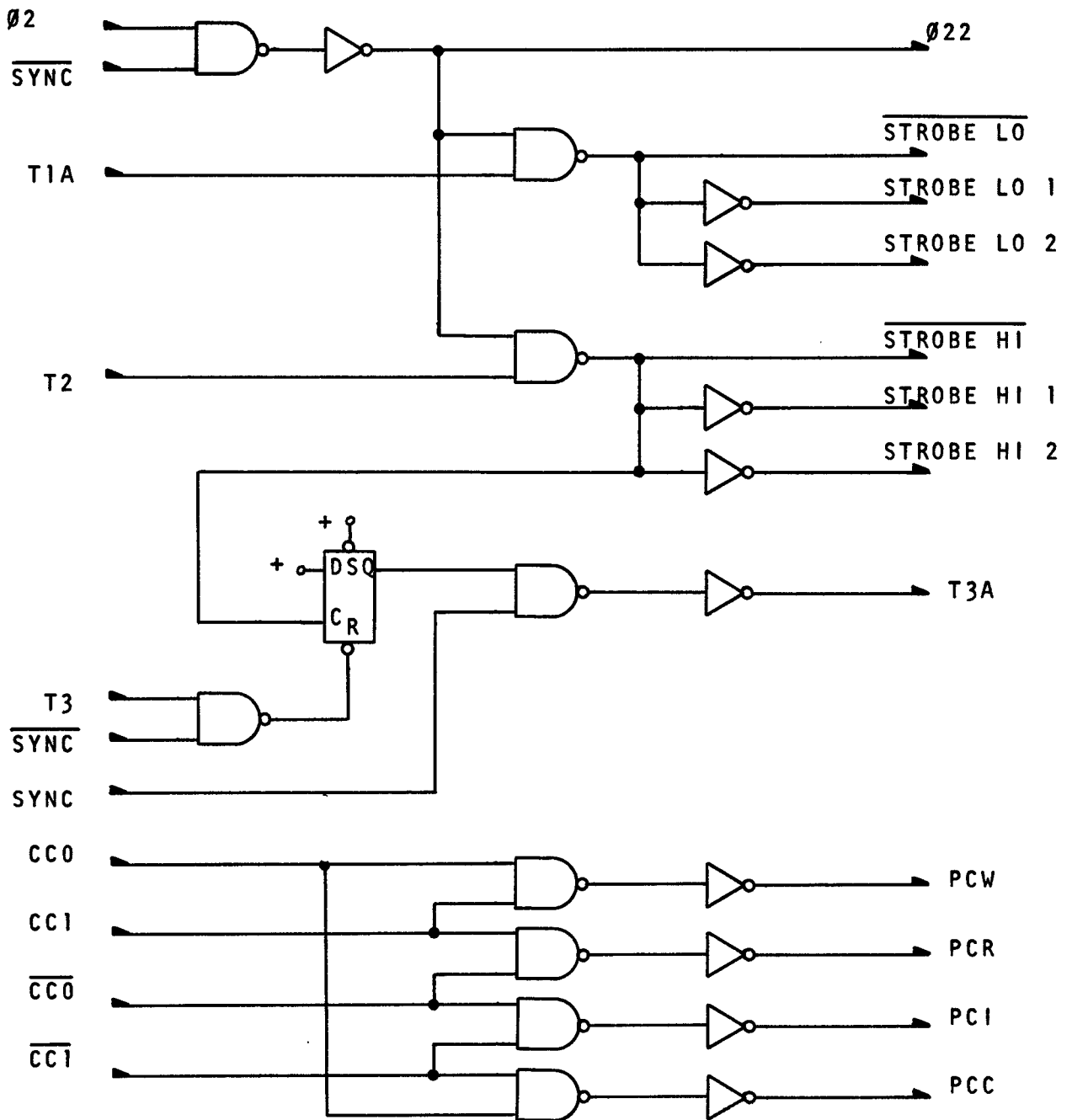


fig.A.2

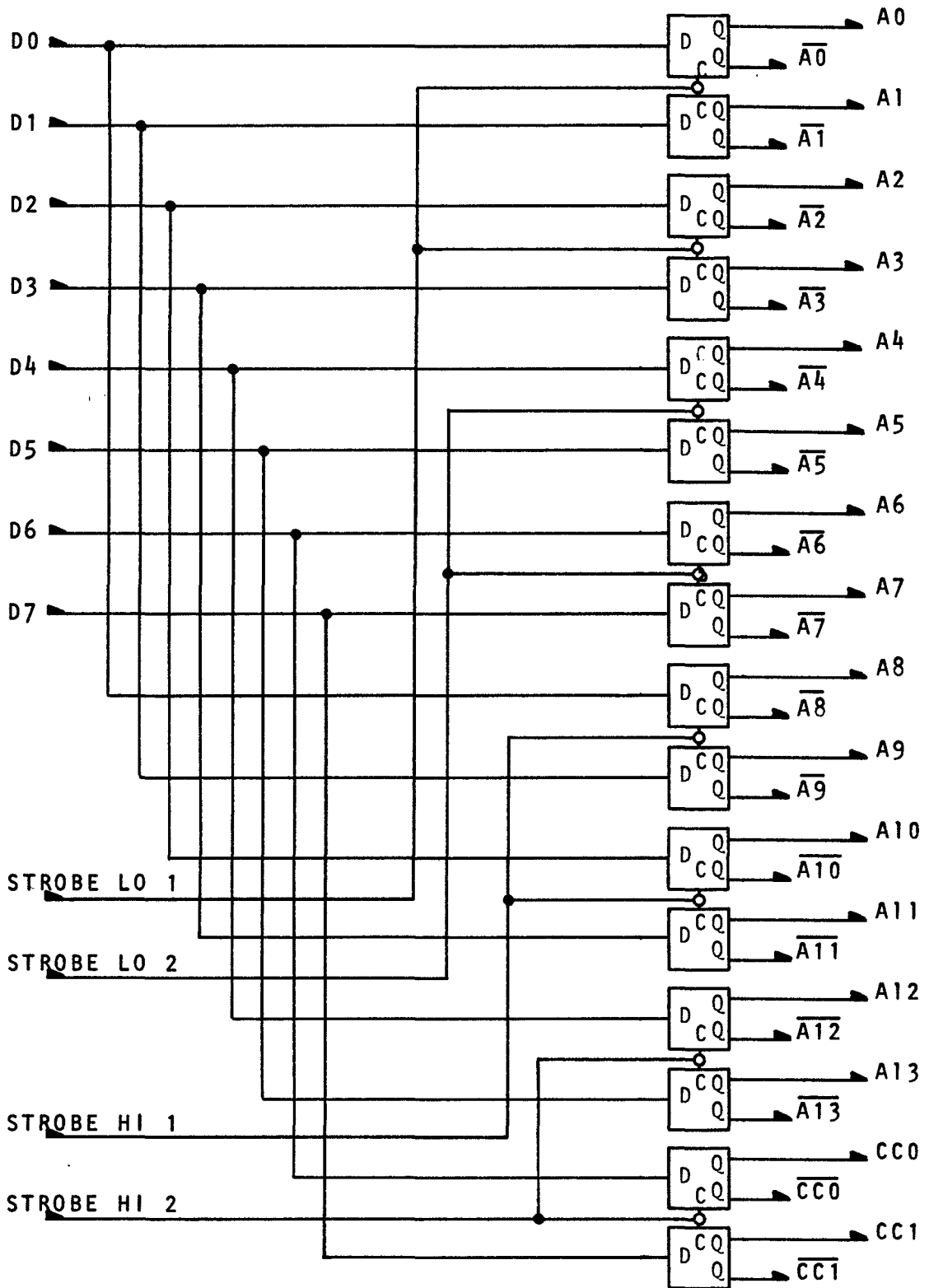


fig A.3

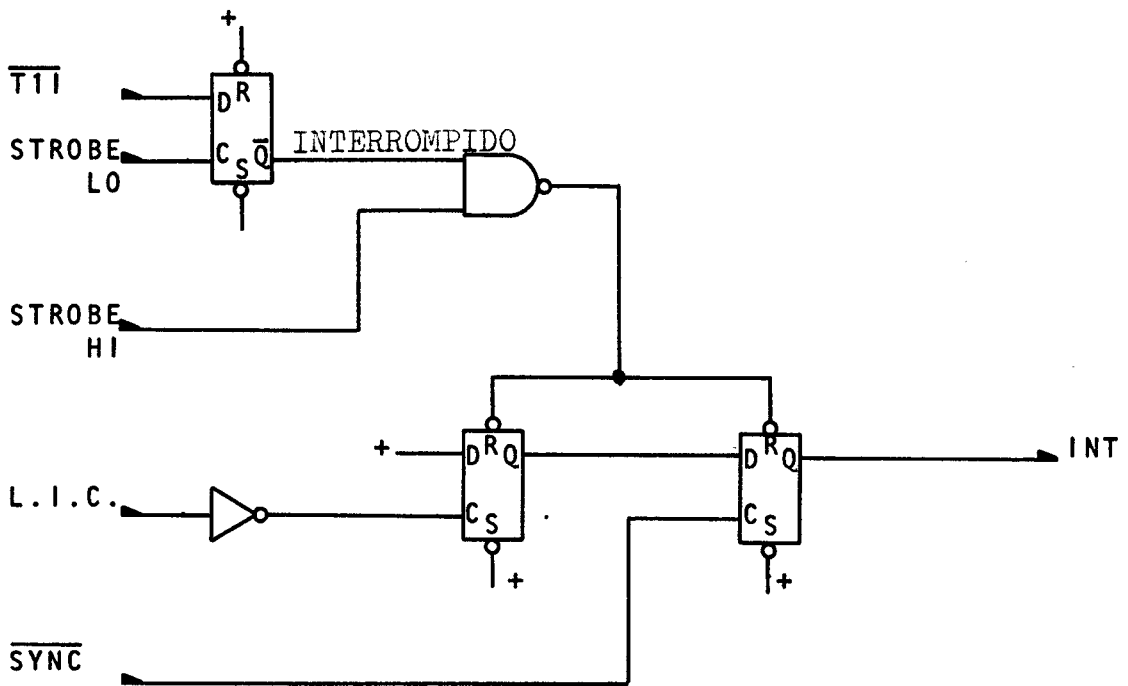
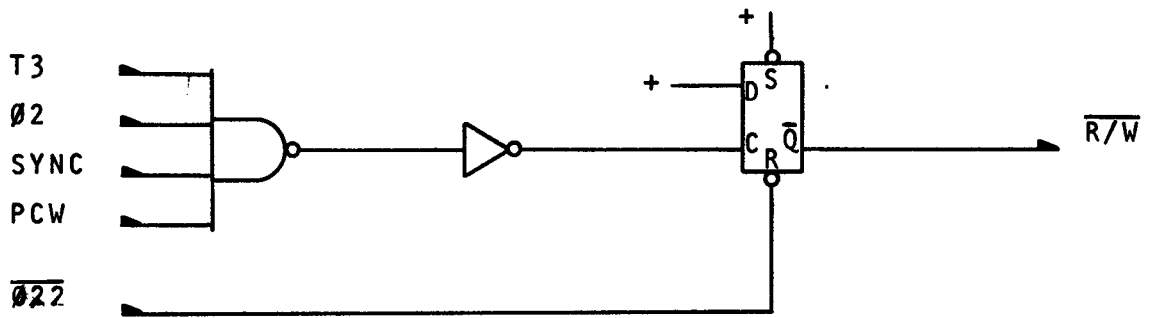


fig A.4

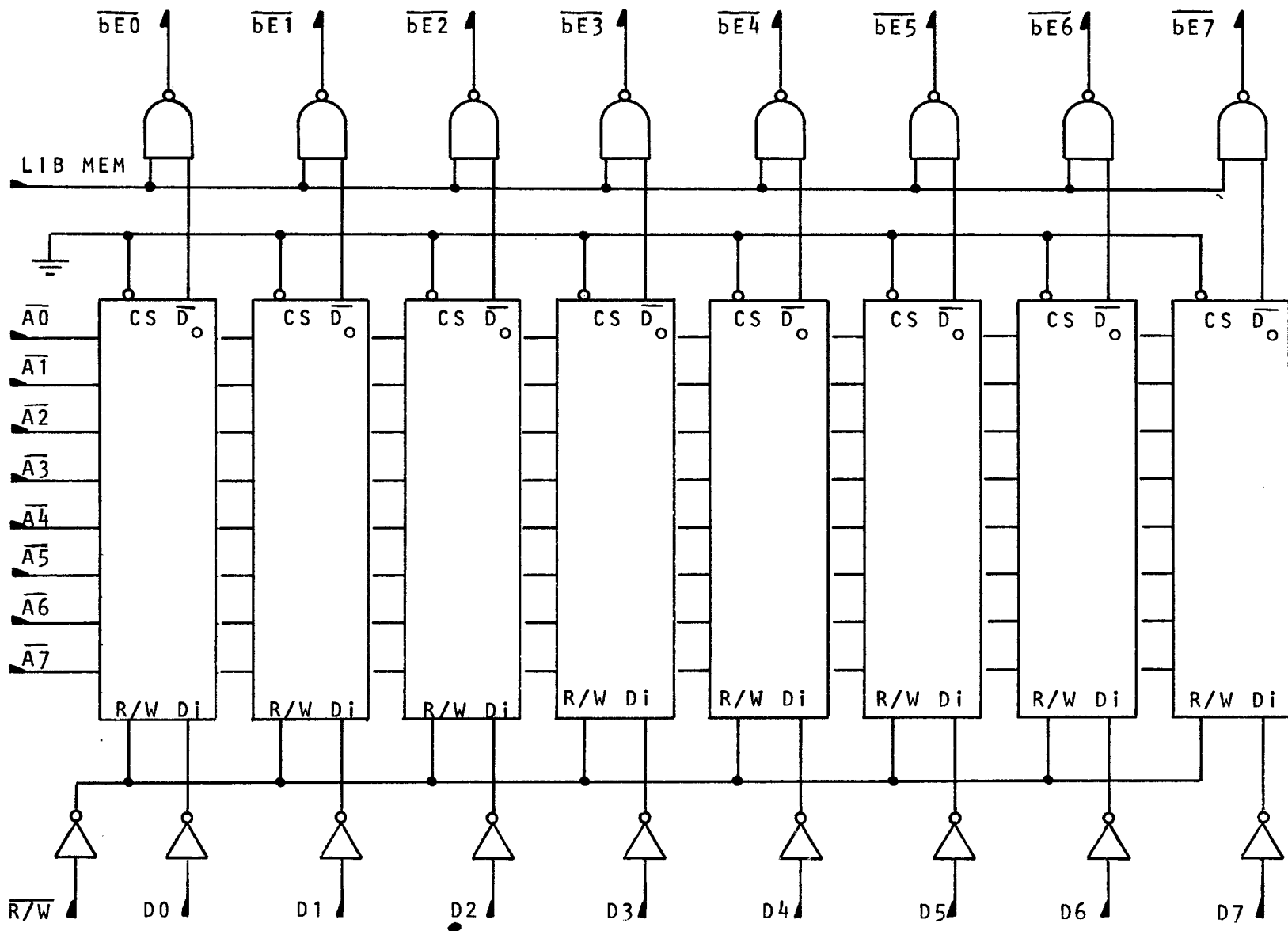
figura A.5 Memória

Oito módulos de 256 x 1 bits (INTEL 1101A).

figura A.6 Periféricos

Registros periféricos 37, 17 e 7. Registro de armazenamento das condições C, Z, S e P. Duas LEDs (Light Emitting Diodes), TI 209, para WAIT e STOP.

fig A.5



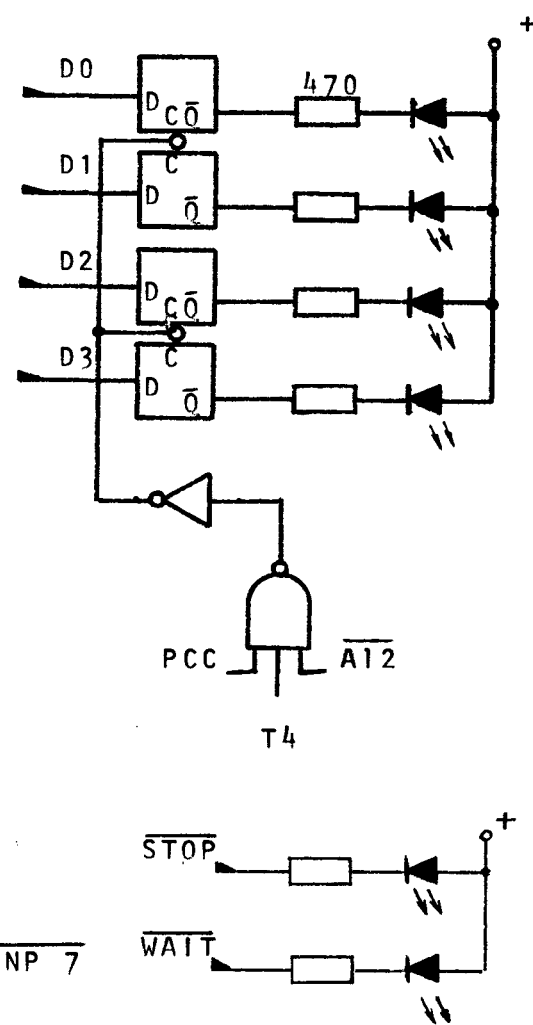
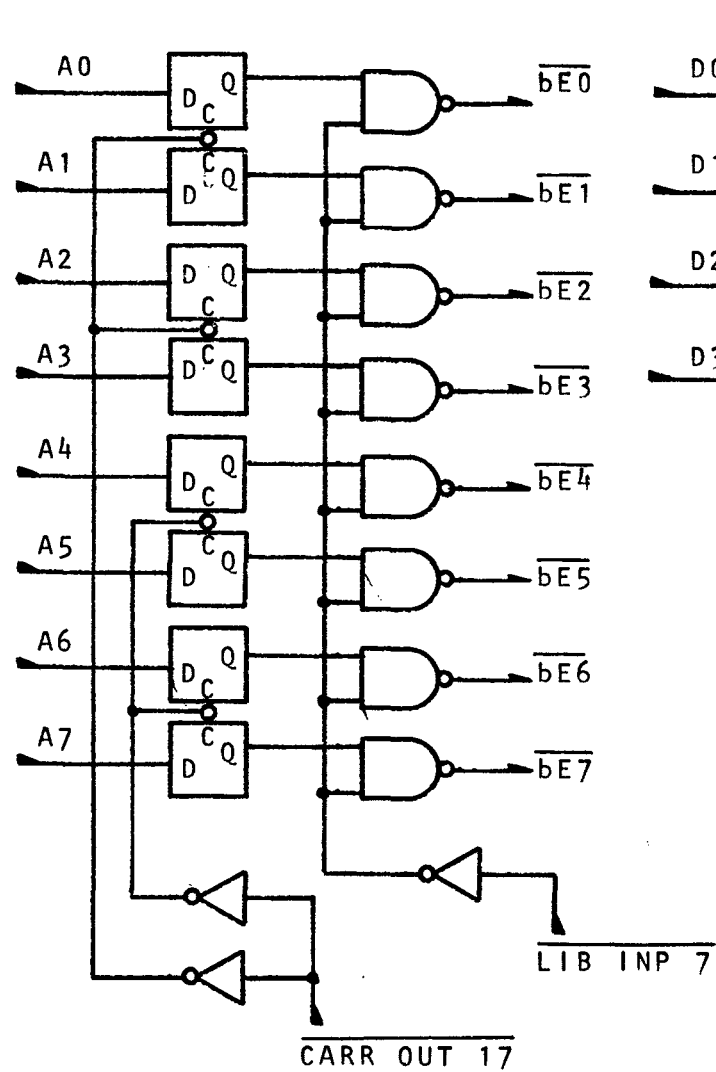
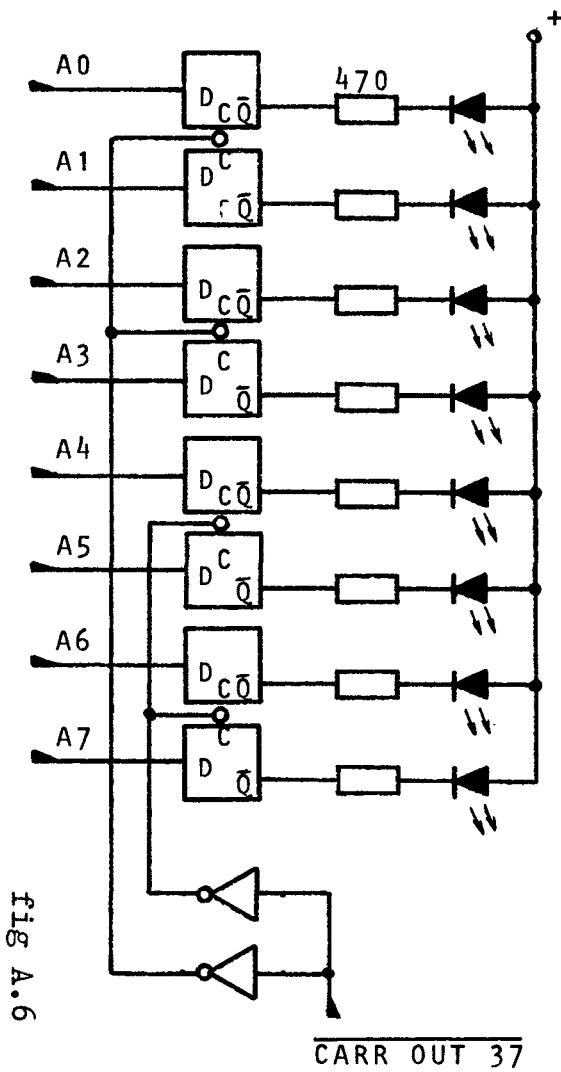


figura A.7 Painel

Tecclas para controle de acesso a memória. Filtragem dos sinais (circuitos "anti-bounce"). Geração do endereço da primeira instrução na MAPRO ($\overline{\text{MEM}}$)

figura A.8 Painel

Tecclas para controle do processador. Filtragem dos sinais. Geração do endereço da primeira instrução ($\overline{\text{CPU}}$, R1, R2, R3).

figura A.9 Painel

Contador de endereço da MAPRO. O sinal gerado com os sinais do processador, é diferenciado por um circuito RC para obtenção de um pulso de 50 ns.

figura A.10 MAPRO

Decodificação do endereço. Matriz de diodos. Circuitos de geração de interrupção.

figura A.11 Painel

Registro RP e lâmpadas para visualização de seu conteúdo.

figura A.12 Painel

Circuito usado na obtenção do segundo octeto das instruções longas. Flip-flop para identificação das instruções de referencia imediata e de referencia à memória sob interrupção.

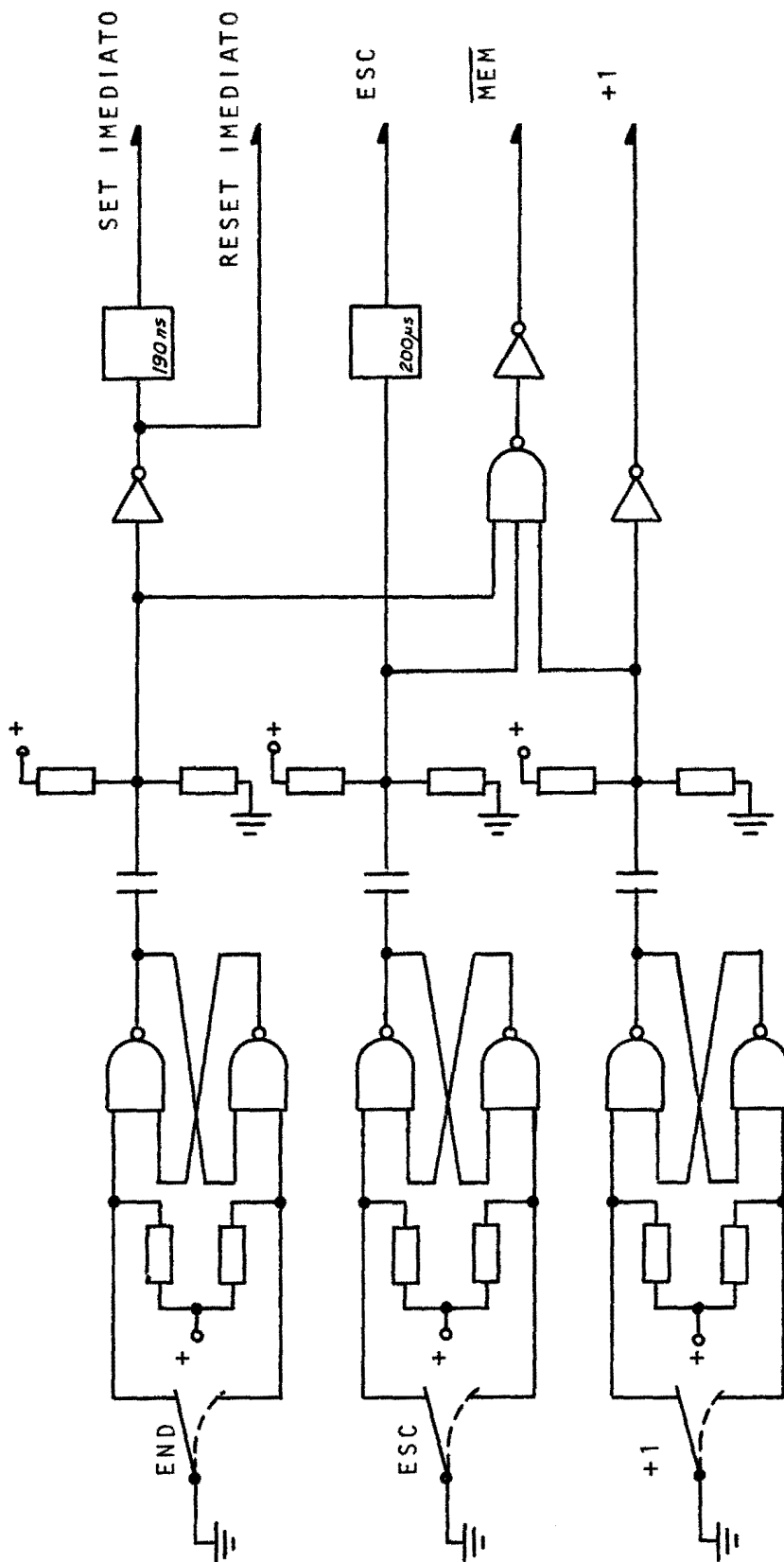


fig A.7

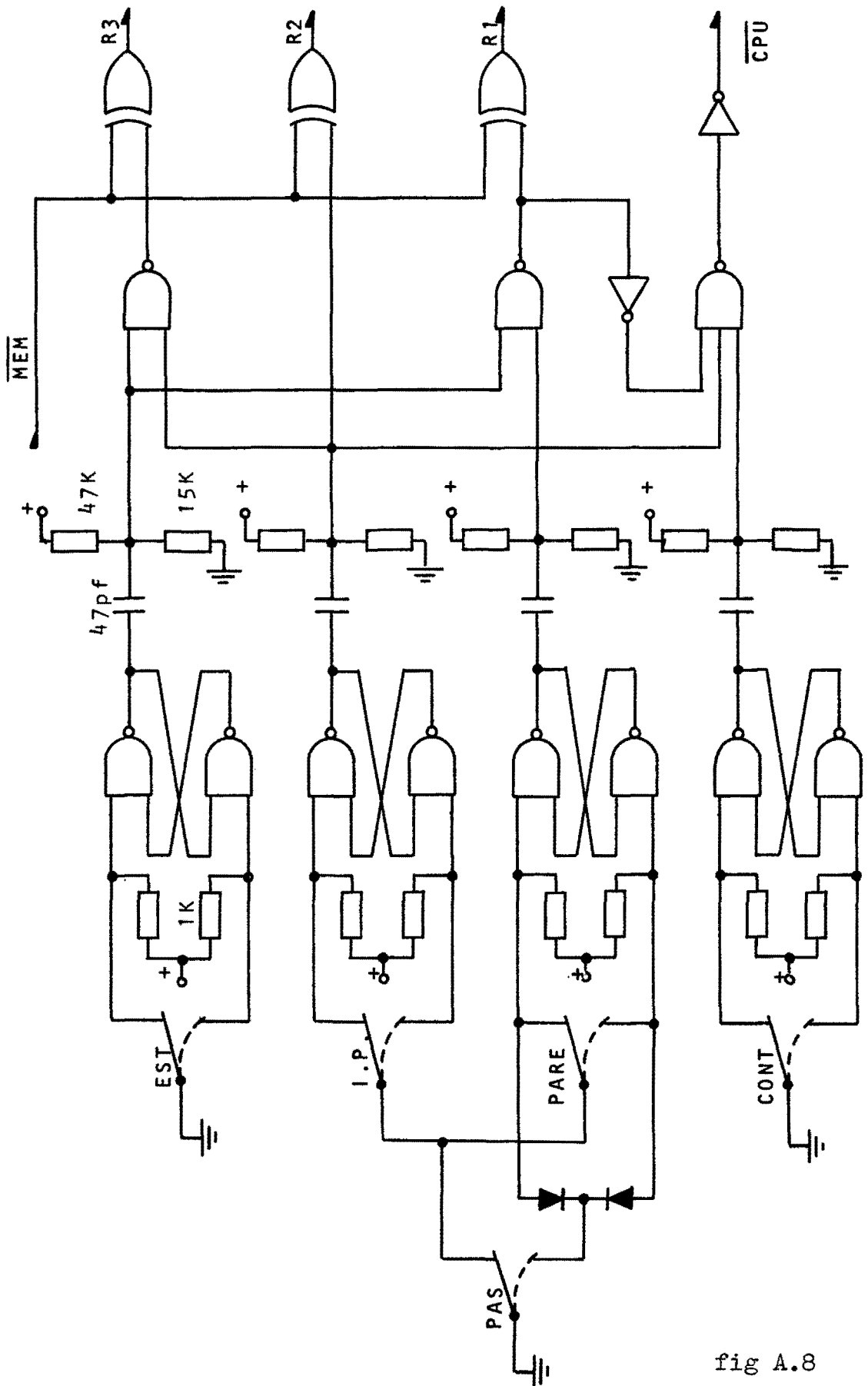


fig A.8

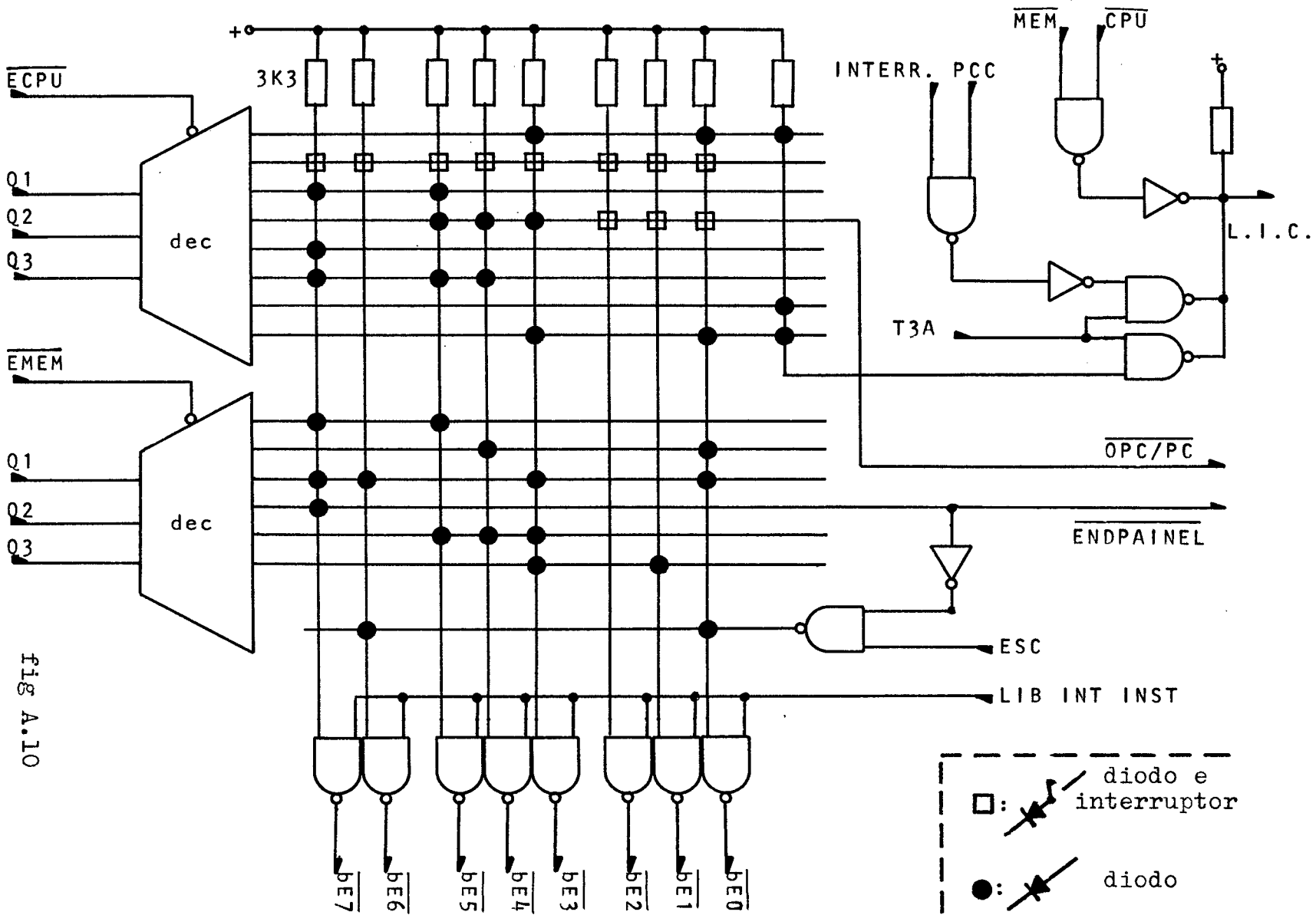


fig A.10

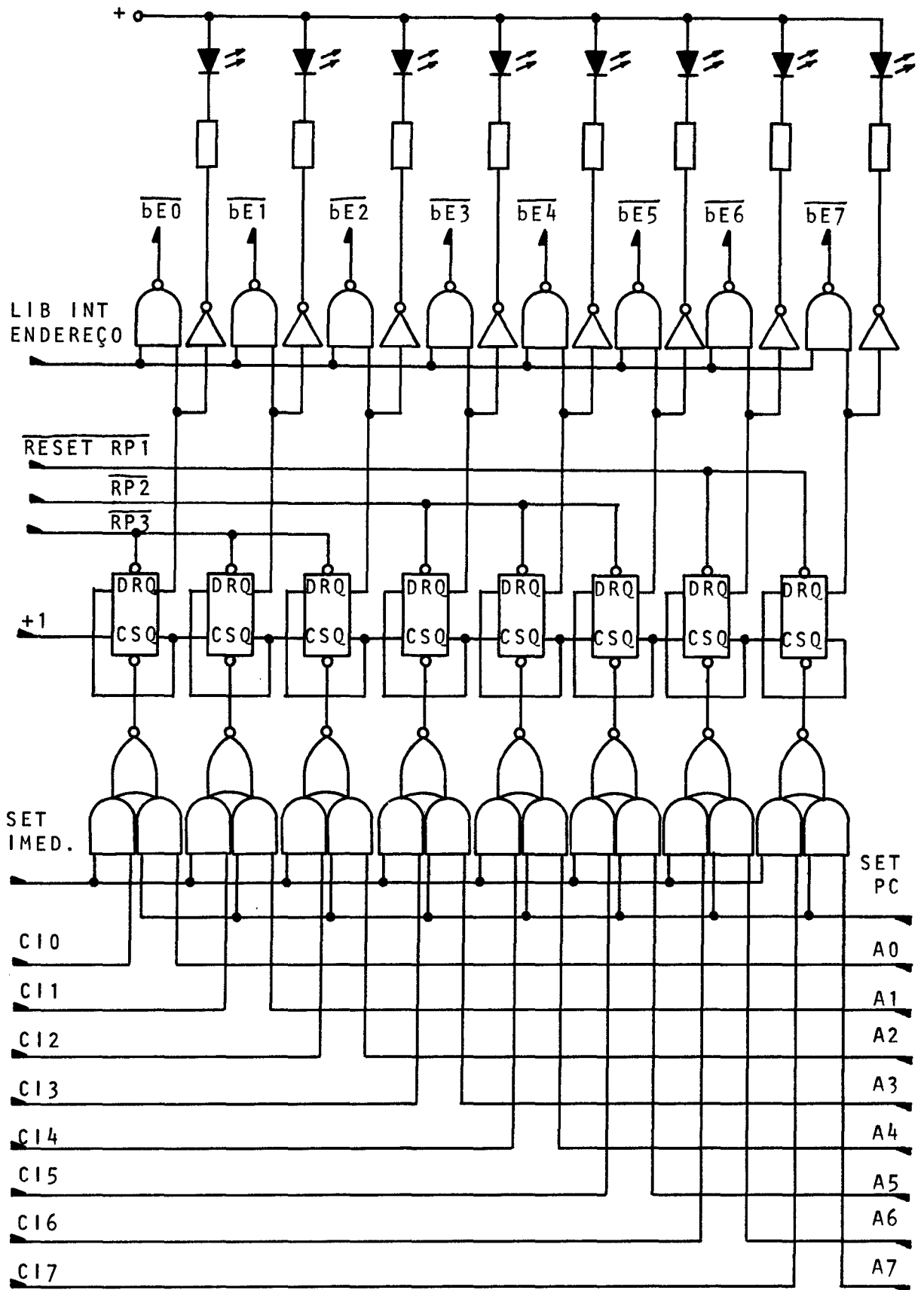


fig A.11

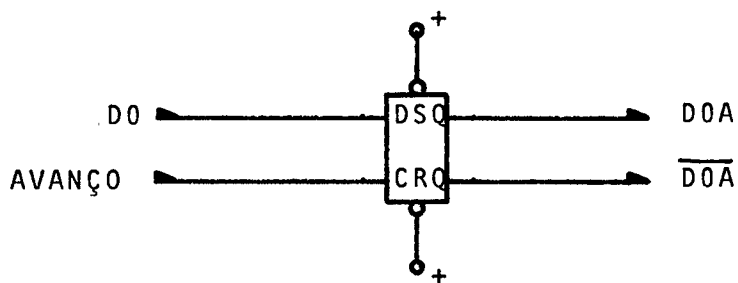
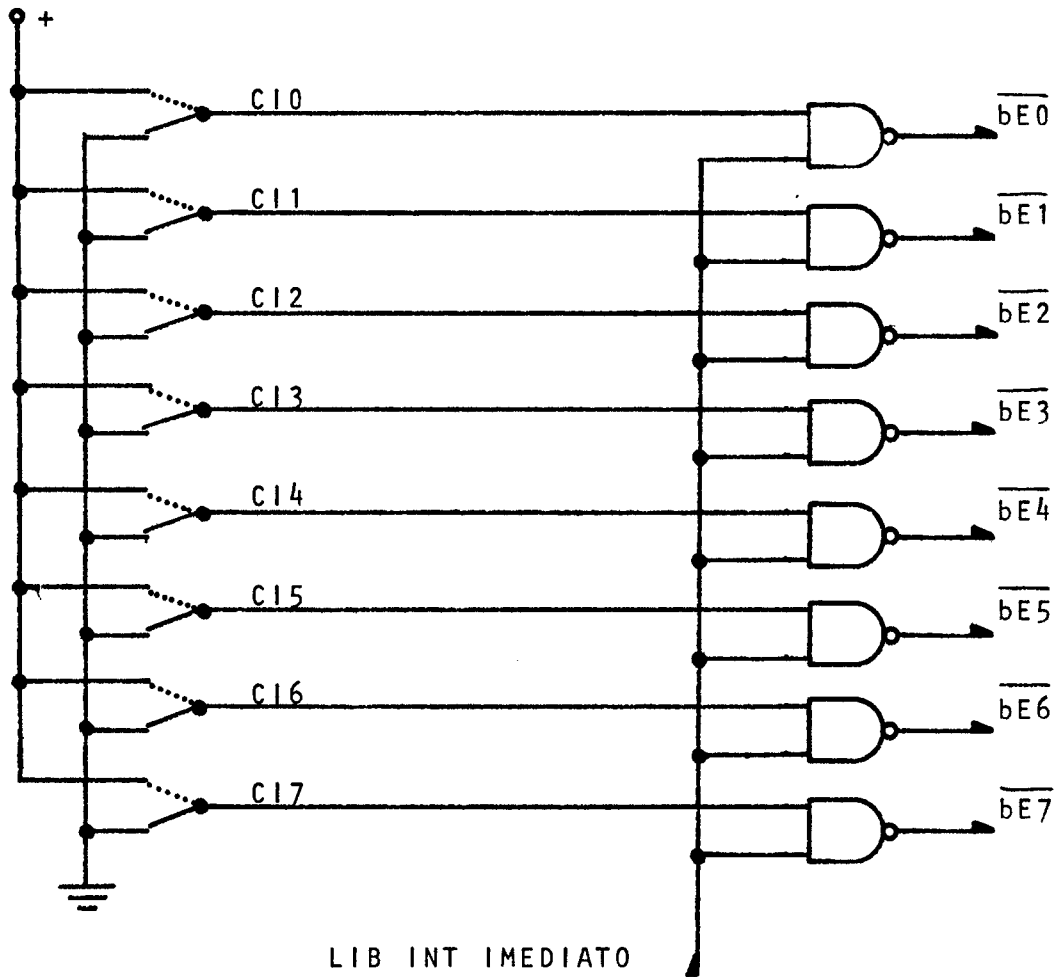


fig A.12

figura A.14 Barra E, painel e periféricos

Circuitos para carregamento dos periféricos, multiplexação da barra E e carregamento de RP.

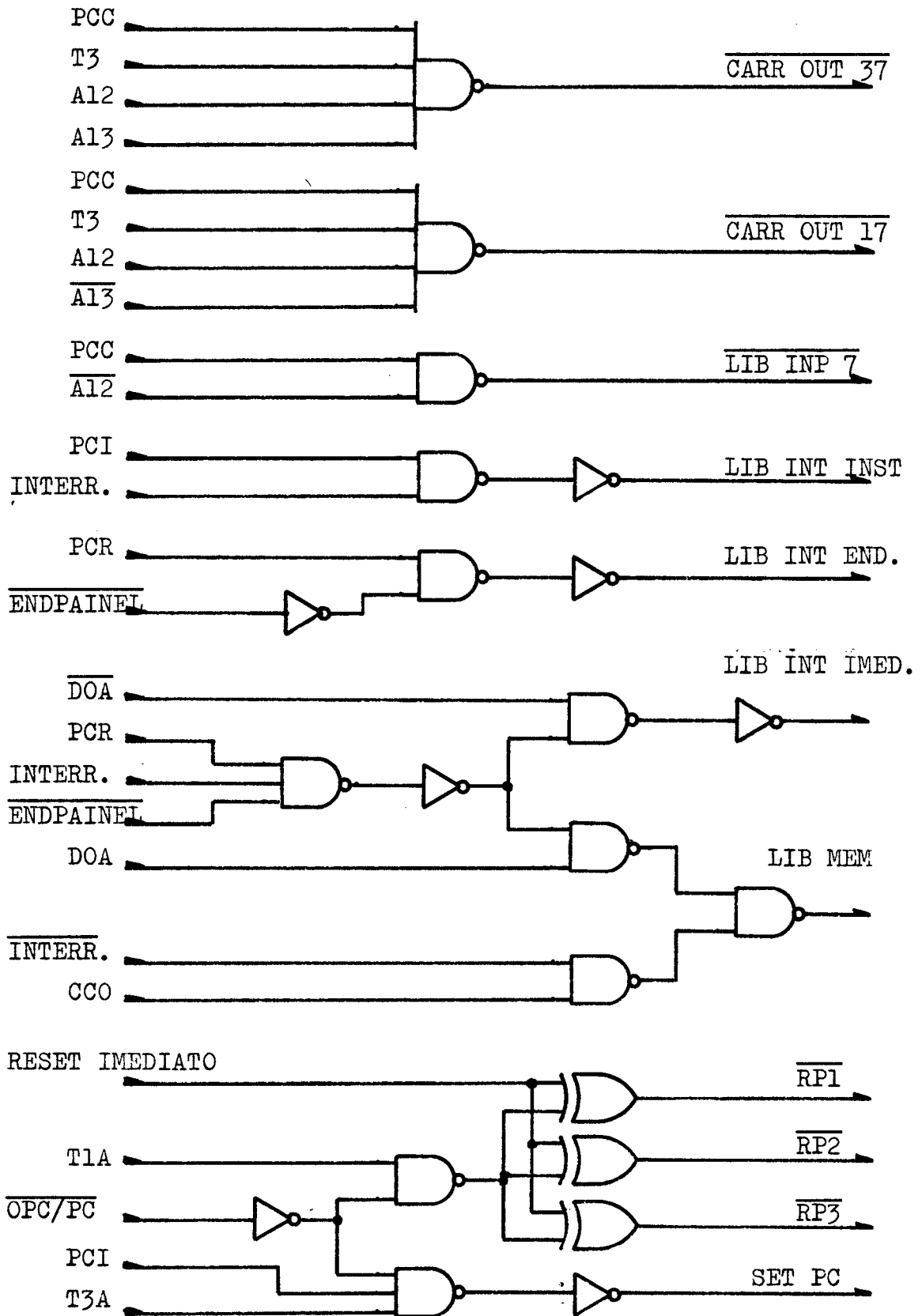


fig A.14

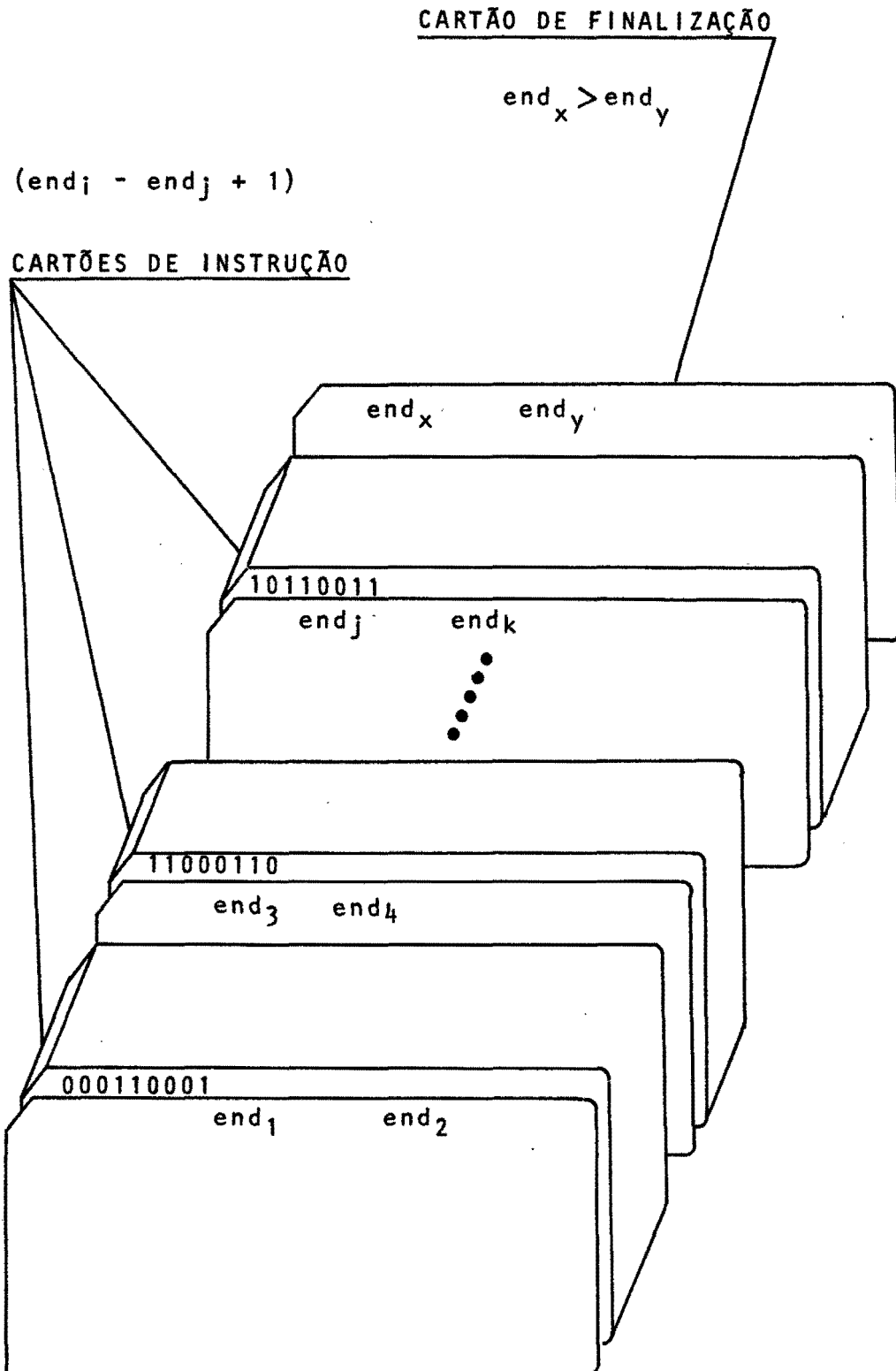
A P E N D I C E B

COMO USAR O SIMULADOR

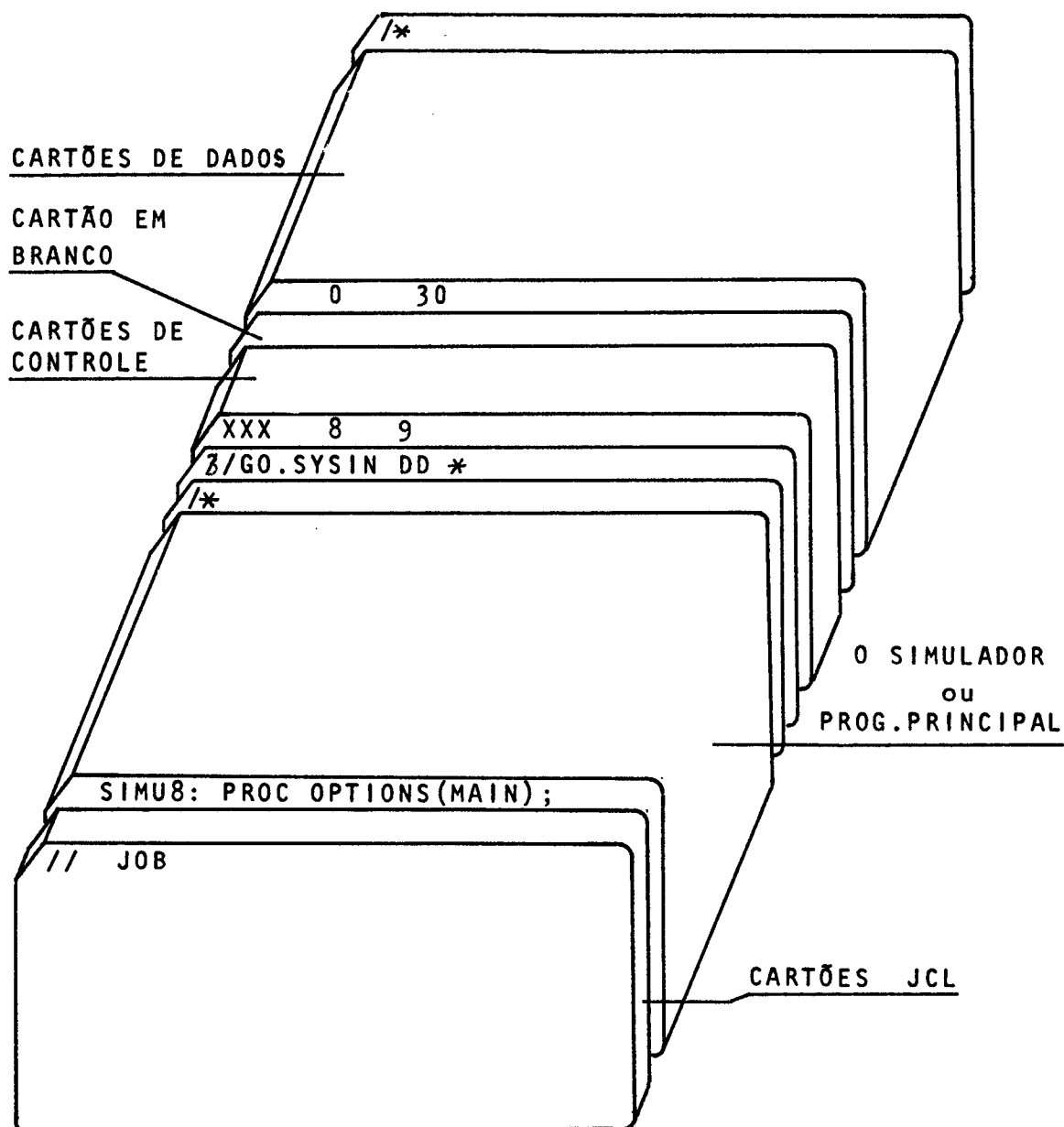
Os dados do simulador serão constituídos por cartões de controle e cartões de dado ou programa. Esses últimos terão o código binário da instrução perfurado nas 8 primeiras colunas do cartão. As colunas de 9 a 72 estão reservadas para algum comentário e as de 73 a 80 para o nome da instrução se o programador desejar. Apenas o código binário é obrigatório.

Os cartões de programa virão precedidos por cartões (sem formato) que irão especificar o primeiro e último endereço onde deve ser carregado o programa em memória.

Os cartões dos programas:



Disposição dos cartões no simulador:



CARTÕES DE CONTROLE	NOME	DEFINE	OBSERVAÇÃO	CARTÕES	"DEFAULT"
Cartão PAG	PAG	tamanho da página	deve vir antes dos cartões tipos	0 a 1	256
Cartões TIPO	ROM RAM EQU ZER UM	pág. ROM pág. RAM pág. equival. a outra pág. só lê 0 pág. só lê 255	podem vir em qual- quer ordem após o cartão PAG	0 a ∞	0 cartões implica em páginas RAM; 1 ou mais implica em acesso pro- ibido às páginas não definidas.
Cartão IMP	IMP	pts. de quebra	pode vir em qual- quer posição	0 a 1	0 a 16384, ie, toda a memória
Cartão BRANCO	xxx	fim dos cartões de controle	é o último car- tão de controle	1	(esse cartão é obri- gatório)

FORMATO DOS CARTÕES DE CONTROLE

<u>coluna</u>	<u>fç.</u>	<u>PAG</u>	<u>ROM</u> <u>ZER</u> <u>UM</u> <u>RAM</u>	<u>EQU</u>	<u>IMP</u>
1 a 10	nome	PAG	ROM ZER UM RAM	EQU	IMP
11 a 15	t1	tam.pág.	tempo leitura		enderêço t1(incl)
16 a 20	te		tempo escrita		ao end. te(excl)
21 a 25	pm(1)		página pm(i) inclusive, a página xm(i) exclusive, onde i varia de 1 a 6	pm(i-1)=pm(i)	end. pm(i) (incl)
26 a 30	xm(1)			⋮	ao end. xm(i)
31 a 35	pm(2)			xm(i-1)-1=pm(i)+k	(exclusive),
36 a 40	xm(2)			pm(i-1)=pm(i)+k+1	onde i varia de 1
41 a 45	pm(3)			⋮	a 6.
46 a 50	xm(3)			pm(i-1)+j=xm(i)-1	
51 a 55	pm(4)			onde i=2,4,6	
56 a 60	xm(4)				
61 a 65	pm(5)				
66 a 70	xm(5)				
71 a 75	pm(6)				
76 a 80	xm(6)				

-A página pode ter qualquer valor que seja uma potência de dois entre $1(2^0)$ e $16384(2^{14})$

-A fç. nome é "left-justified"; as outras são "right-justified".

A P E N D I C E C

LISTAGENS

Programa em linguagem PL/1 para a simulação de um micro-computador que usa a 8008.

Este programa exige de 90 a 100 K de memória.

SIMU8: PROCEDURE OPTIONS (MAIN);	SIMU0000
ON ENDFILE(SYSIN) GOTO GRAN_FINAL;	SIMU0005
NOVO_PROGRAMA: BEGIN;	SIMU0006
ON ENDPAGE (SYSPRINT) BEGIN; PUT PAGE; PUT LINE(30);	
IF FIM_DA_PARTE_INICIAL THEN	
PUT EDIT ('IPC PC INST C Z S P A B C D E H L M',	SIMU0011
'ESTADOS')(COL(19),A,COL(97),A); FND;	SIMU0012

```

/*****/SIMU0100
/* DECLARACAO E INICIALIZACAO DAS VARIABEIS PARA 8008 */SIMU0101
/*****/SIMU0102
DCL /* ELEMENTOS DE MEMORIA DO MICRO PROCESSADOR SIMU0104
      FLIP FLOPS DE CONDICAO: FFCOND(0)=CARRY=BORROW SIMU0106
                                1 ZERO SIMU0108
                                2 SYGN SIMU0110
                                3 PRTY SIMU0112
      FF DE INTERRUPCAO : INTERRUPT SIMU0114
      REGISTROS DE INDEXACAO: REGINDEX(0)=REGISTRO A SIMU0116
                                1 B SIMU0118
                                2 C SIMU0120
                                3 D SIMU0122
                                4 E SIMU0124
                                5 H SIMU0126
                                6 L SIMU0128
      ACUMULADOR : ACUMULADOR=REGINDEX(0) SIMU0132
      CONTADORES DE PROGRAMA: PC SIMU0134
      REGISTROS TEMPORARIOS : REGA E REGB SIMU0136
      REGISTRO DE INSTRUCAO : REGI SIMU0138
      APONTADOR DE PC : IPC SIMU0140
      APONTADOR DE REGINDEX : IREGIND */ SIMU0142
      (FFCOND(0:3)) FIXED BIN(1) ,SIMU0144
      CARRY(1) FIXED BIN(1) DEF FFCOND(1SUB-1) ,SIMU0146
      BORROW(1) FIXED BIN(1) DEF FFCOND(1SUB-1) ,SIMU0148
      ZERO(1) FIXED BIN(1) DEF FFCOND(1SUB) ,SIMU0150
      SYGN(1) FIXED BIN(1) DEF FFCOND(1SUB+1) ,SIMU0152
      PRTY(1) FIXED BIN(1) DEF FFCOND(1SUB+2) ,SIMU0154
      INTERRUPT FIXED BIN(1) INIT(0) ,SIMU0156
      REGINDEX(0:6) FIXED BIN(8) INIT((7)0) ,SIMU0158
      ACUMULADOR(1) FIXED BIN(8) DEF REGINDEX(1SUB-1) ,SIMU0160
      PC(0:7) FIXED BIN(16) INIT((8)0) ,SIMU0162
      (REGA,REGB) FIXED BIN(8) INIT(0) ,SIMU0164
      REGI BIT(8) ,SIMU0166
      (IPC,IREGIND) FIXED BIN(3) INIT(0) ;SIMU0168

```

```

DCL      /* VARIAVEIS REFERENTES A MEMORIA EXTERNA AO PROCESSADOR I8008 SIMU0200
          TAMANHO DA PAGINA      : PAG                      SIMU0202
          DEFINICAO DA MEMORIA:  IMAGEM_MEMORIA            SIMU0201
          ROM, RAM, ...          : TIPO                     SIMU0206
          "BOOTSTRAP"           : CARREGAMENTO             SIMU0208
          TEMPOS DE ACESSO A     : TEMPO_LEITURA          SIMU0210
          CADA MODULO            TEMPO_ESCRITA             SIMU0212
          MEMORIA REAL           : MEM                      SIMU0214
          ENDERECO PARA MEM      : IMEM                    */SIMU0216
          PAG                    FIXED BIN(16)      INIT(256) ,SIMU0218
          M                      FIXED BIN(16)      DEF      PAG ,SIMU0220
1  IMAGEM_MEMORIA(*)                                CTL      ,SIMU0222
2  TIPO                      CHAR(3)                ,SIMU0224
2  CARREGAMENTO              BIT(1)                 ,SIMU0226
2  TEMPO_LEITURA            FIXED DEC(5)            ,SIMU0228
2  TEMPO_ESCRITA             FIXED DEC(5)            ,SIMU0230
          MEM(0:16383)        BIT(8)                ,SIMU0232
          IMEM                FIXED DEC(3)          INIT(0)  ;SIMU0234

DCL      /* VARIAVEIS AUXILIARES PARA CARREGAMENTO DOS PROGRAMAS----- SIMU0250
          DEFINICAO DO TIPO DE MEMORIA      : CONTROLE      SIMU0252
          LIMITES DAS PARTES DE UM PROGRAMA: INICIAL E FINAL SIMU0254
          INSTRUCAO LIDA                    : INSTRUCAO      SIMU0256
          OBSERVACOES DO PROGRAMADOR        : COMENTARIO     */SIMU0258
          (INICIAL,FINAL)    FIXED DEC(3)  INIT(0)          ,SIMU0260
          CONTROLE           CHAR(3)                ,SIMU0262
          (TL,TE)            FIXED DEC(5)            ,SIMU0264
          (PM(6),XM(6))      FIXED BIN(16)           ,SIMU0266

          COMENTARIO         CHAR(64)                ,SIMU0268

          INSTRUCAO          CHAR(8)                  ;SIMU0270

```



```

DCL      /* VARIAVEIS PARA IMPRESSAO DOS RESULTADOS                                SIMU0300
          A INSTRUCAO PROCESSADA                : NOME_DA_INSTRUCAO                SIMU0302
          IMEDIATO, MEMORIA OU REGISTRO          : TIPO_DE_REFERENCIA              SIMU0304
          O REGISTRO USADO                      : NOME_DO_REGISTRO                 SIMU0306
          O "FLIP-FLOP" TESTADO                 : NOME_DO_FF                          SIMU0308
          DIZ QUANDO HOUVE SALTO                SIMU0310
          O QUE FOI ESCRITO NA MEM              SIMU0312
          DADO DE ENTRADA                      SIMU0314
          TENTATIVA DE ESCRITA EM ROM           : OBSERVACAO                          SIMU0316
          IMEDIATO OU MEMORIA                  : OCTETO                             SIMU0318
          TESTE SE 1 OU 0                      : TRUE_FALSE                         SIMU0320
          "BREAKPOINTS"                       : TRACE_ON      TRACE_OFF           SIMU0322
          QUANDO IMPRIMIR INSTRUCAO            : IMPRIMA                           SIMU0324
          QUANDO IMPRIMIR CABECALHO            : FIM_DA PARTE_INICIAL              */SIMU0326

```

```

NOME_DA_INSTRUCAO      CHAR(6)  INIT('      ')      ,SIMU0328
TIPO_DE_REFERENCIA     CHAR(1)  INIT(' ')           ,SIMU0330
NOME_DO_REGISTRO(0:6)  CHAR(1)  INIT('A','B','C','D','E','H','L'),SIMU0332
NOME_DO_FF(0:3)        CHAR(1)  INIT('C','Z','S','P') ,SIMU0334
(OBSERVACAO,OCTETO)    CHAR(8)  INIT('      ')      ,SIMU0336
TRUE_FALSE             CHAR(1)  ,SIMU0338
TRACE_ON(7)           FIXED BIN(16) INIT((7)0)      ,SIMU0340
TRACE_OFF(7)          FIXED BIN(16) INIT((7)16384)   ,SIMU0342
IMPRIMA               BIT(1)    INIT('1'B)           ,SIMU0344
FIM_DA PARTE_INICIAL  BIT(1)    INIT('0'B)           ;SIMU0346

```

```

DCL      /* VARIÁVEIS PARA CONTROLE DE FLUXO ENTRE MÓDULOS                SIMU0350
          DECODIFICAÇÃO INICIAL DA INSTRUÇÃO : DECODIFICAÇÃO_DE_D7D6      SIMU0352
          (A DECODIFICAÇÃO DE D5D4D3 FEITA POR TESTES)                     SIMU0354
                                          DECODIFICAÇÃO_DE_D2D1D0SIMU0356
          DECODIFICAÇÃO FINAL DA INSTRUÇÃO   : LABEL_ROTATE                SIMU0358
                                          LABEL_JUMP_CALL                  SIMU0360
          SELEÇÃO DO PERIFÉRICO              : MÓDULO_INP                  SIMU0362
                                          MÓDULO_OUT                        SIMU0364
          INSTRUÇÃO SOB INTERRUPÇÃO                                                SIMU0366
          PRIMEIRA PALAVRA(INSTRUÇÃO) : INT_PCI                          SIMU0368
          SEGUNDA PALAVRA(IMEDIATO OU : INT_PCR_I                        SIMU0370
          TERCEIRA PALAVRA(ENDERECO) : INT_PCR_MM                      SIMU0372
          SEMÁFORO IDENTIFICADOR DE INSTRUÇÕES                                SIMU0374
          COM REFERÊNCIA 'IMEDIATO'   : LABEL1 (=MÓDULO_ALU OU          SIMU0376
                                          =MÓDULO_LR_MI)SIMU0378
          COM REFERÊNCIA 'MEMÓRIA'    : LABEL2 (=MÓDULO_ALU OU          SIMU0380
                                          =MÓDULO_LRM)SIMU0382
          CONTROLE P/ "ALLOCATE"      : LABEL3                          SIMU0384
          ÍNDICES PARA DECODIFICAÇÃO INICIAL : D7D6,D5D4D3,D2D1D0      SIMU0386
          QUAL O PERIFÉRICO           : VALOR_PERIFÉRICO                SIMU0388
          SELEÇÃO DA INTERRUPÇÃO      : VALOR_DA_INTERRUPÇÃO */SIMU0390

(DECODIFICAÇÃO_DE_D7D6(0:3),DECODIFICAÇÃO_DE_D2D1D0(0:7),                SIMU0392
 LABEL_ROTATE(0:3),LABEL_JUMP_CALL(0:3),                                SIMU0394
 MÓDULO_INP(0:7),MÓDULO_OUT(10:37),                                     SIMU0396
 LABEL1,LABEL2,LABEL3) LABEL,                                          SIMU0398
(INT_PCI( 5),INT_PCR_I( 5),INT_PCR_MM( 5)) LABEL ,                     SIMU0400
(D7D6,D5D4D3,D2D1D0) FIXED BIN(3) ,                                  SIMU0402
(VALOR_PERIFÉRICO,VALOR_DA_INTERRUPÇÃO) FIXED DEC(2) ;                SIMU0404

```

```

DCL.  /* VARIAVEIS AUXILIARES                                SIMU0450
      RFLOGIO "TEMPO REAL" : NUMERO_DE_ESTADOS                SIMU0452
      CALCULO DO ENDERFLO  : FND_MEM                          SIMU0454
      CALCULO DO MODULO    : MOD_MEM                           SIMU0456
      "BUFFER" DE SAIDA    : OUT_LATCH                         SIMU0458
      (NUMERO_DE_ESTADOS*4 FORNECE TEMPO EM MICROSEGUNDOS)    SIMU0460
      TESTE 0 OU 1 DOS FF  : CONDICAO                          SIMU0462
      PARA CALCULO DOS FF  : REGTESTE                          SIMU0464
      PARA CALCULO DE PRY  : NN                                SIMU0466
      CONTADOR                                                       SIMU0468
      PARA CALCULO DO CARRY: UNDERFLOW                         SIMU0470
      OVERFLOW                                                       SIMU0472
      CARRY PARA A ROTACAO : CARRYAUX                          */SIMU0474

      NUMERO_DE_ESTADOS      FIXED      DEC(15)      INIT(0),      SIMU0476
      (END_MEM,PAG_MEM)      FIXED      BIN(16) ,      SIMU0478
      MOD_MEM                FIXED      BIN(16)      DEF      PAG_MEM ,
      OVERFLOW(1)            FIXED      BIN( 1)      INIT(0B),      SIMU0480
      UNDERFLOW(1)          FIXED      BIN( 1)      DEF      OVERFLOW , SIMU0482
      (CONDICAO,CARRYAUX(1)) FIXED      BIN( 1)      ,      SIMU0484
      (REGTESTE,OUT_LATCH)   FIXED      BIN( 8)      ,      SIMU0486
      (NNN,CONTADOR)         FIXED      BIN( 5)      ;      SIMU0488

/* VARIAVEIS DEFINIDAS OFLO USUARIO */                      SIMU0498
DECLARE DADO_DE_ENTRADA(0:7) FIXED BINARY(8) INITIAL((8)0),    SIMU0490
      FFLATCH FIXED BIN(8);                                       SIMU0492

DCL STOPPED FIXED BIN(16) INIT(0);
DCL GUARDA_ACUMULADOR FIXED BIN(8);                               SIMU0494

```

```

/***** */SIMU1000
/*** M O D U L O   L E I T U R A   C A R T A O   C O N T R O L E *** */SIMU1001
/***** */SIMU1002
LEITURA_CARTAO_CONTROLE: GET EDIT(CONTROLE)(COL(1),A(3)) COPY;          SIMU1004

IF CONTROLE='PAG' THEN DO;          SIMU1006
  GET EDIT(PAG)(COL(11),F(5)) COPY;
  IF PAG>16384 THEN PUT LIST ('PAGINA ULTRAPASSA LIMITE',PAG);          SIMU1010
  IF PAG=1|PAG=2|PAG=4|PAG=8|PAG=16|PAG=32|PAG=64|PAG=128|PAG=256|          SIMU1012
    PAG=512|PAG=1024|PAG=2048|PAG=4096|PAG=8192|PAG=16384 THEN          SIMU1014
    GOTO LEITURA_CARTAO_CONTROLE;          SIMU1016
  ELSE DO; PUT LIST ('PAGINA DEFINIDA NAO FOI UMA POTENCIA DE 2');          SIMU1018
    GOTO ENDE;          FND;          SIMU1020
GOTO LEITURA_CARTAO_CONTROLE;          END;          SIMU1024

IF CONTROLE='EQU' THEN DO;          SIMU1030
  IF -ALLOCATION(IMAGE_MEMORIA) THEN DO;          SIMU1032
    LABEL3=EQU;          SIMU1034
    GOTO IMAGE_INICIAL;          END;          SIMU1036
    EQU: GET EDIT ((PM(I),XM(I) DO I=1 TO 6))(COL(21),12 F(5)) COPY;          SIMU1038
    DO I=2,4,6; IF PM(I)>=XM(I)|PM(I-1)>=XM(I-1) THEN GOTO LOOP10;          SIMU1040
      DO J=0 TO 16384 WHILE((PM(I)+J)-=XM(I)); TIPO(PM(I)+J)='EQU';          SIMU1042
      TEMPO_LEITURA(PM(I)+J)=PM(I-1)+MOD(J,XM(I-1)-PM(I-1) );          SIMU1044
    END;          SIMU1046
  LOOP10: END;          SIMU1048
GOTO LEITURA_CARTAO_CONTROLE;          FND;          SIMU1048

```

IF CONTROLE='ROM' CONTROLE='RAM' THEN DO;	SIMU1052
IF ¬ALLOCATION(IMAGEM_MEMORIA) THEN DO;	SIMU1054
LABEL3=ROM_RAM; GOTO IMAGEM_INICIAL; END;	SIMU1056
ROM_RAM:	SIMU1060
GET EDIT(TL,TE,(PM(I),XM(I) DO I=1 TO 6))(COL(11),14 F(5)) COPY;	SIMU1062
DO I=1 TO 6; IF PM(I)>=XM(I) THEN GOTO LOOP_ROM_RAM;	SIMU1064
DO WHILE(PM(I)¬=XM(I)); TIPO(PM(I))=CONTROLE;	SIMU1066
TEMPO_LEITURA(PM(I))=TL; TEMPO_ESCRITA(PM(I))=TE;	SIMU1068
CARREGAMENTO(PM(I))='1'B; PM(I)=PM(I)+1; END;	SIMU1070
LOOP_ROM_RAM: END;	SIMU1072
GOTO LEITURA_CARTAO_CONTROLE; END;	SIMU1084
IF CONTROLE='ZER' CONTROLE='UM' THEN DO;	SIMU1086
IF CONTROLE='ZER' THEN REGI='00000000'B; ELSE REGI='11111111'B;	SIMU1088
IF ¬ALLOCATION(IMAGEM_MEMORIA) THEN DO;	SIMU1090
LABEL3=ZER_UM; GOTO IMAGEM_INICIAL; END;	SIMU1092
GOTO IMAGEM_INICIAL;	SIMU1094
ZER_UM:	SIMU1096
GET EDIT(TL,TE,(PM(I),XM(I) DO I=1 TO 6))(COL(11),14 F(5)) COPY;	SIMU1098
DO I=1 TO 6; IF PM(I)>=XM(I) THEN GOTO LOOP_ZER_UM;	SIMU1100
DO WHILE(PM(I)¬=XM(I)); TIPO(PM(I))='ROM';	SIMU1102
TEMPO_LEITURA(PM(I))=TL; TEMPO_ESCRITA(PM(I))=TE;	SIMU1104
DO IMEM=PM(I)*M TO PM(I)*(M+1)-1;	SIMU1112
MEM(IMEM)=REGI; END;	SIMU1114
CARREGAMENTO(PM(I))='0'B; PM(I)=PM(I)+1; END;	SIMU1116
LOOP_ZER_UM: END;	SIMU1118
GOTO LEITURA_CARTAO_CONTROLE; END;	SIMU1120
IF CONTROLE='IMP' THEN DO;	SIMU1122
GET EDIT((TRACE_ON(I),TRACE_OFF(I) DO I=1 TO 7))(COL(11),14 F(5))	SIMU1124
COPY;	SIMU1126
GOTO LEITURA_CARTAO_CONTROLE; END;	SIMU1128

IF CONTROLE=' ' THEN GOTO MAQUINA_DEFINIDA;	SIMU1140
PUT LIST('CONTROLE NAO IDENTIFICADO');	SIMU1142
GOTO FNDE;	SIMU1144
IMAGEM_INICIAL: ALLOCATE IMAGEM_MEMORIA(0:16384/M-1);	SIMU1146
DO I=0 TO 16384/M-1;	SIMU1148
TIPO(I)=' ';	SIMU1150
CARREGAMENTO(I)='0'B;	SIMU1152
TEMPO_LEITURA(I),TEMPO_ESCRITA(I)=0;	SIMU1154
END;	SIMU1156
GOTO LABEL3;	SIMU1158
MAQUINA_DEFINIDA: IF ~ALLOCATION(IMAGEM_MEMORIA) THEN DO;	SIMU1170
ALLOCATE IMAGEM_MEMORIA(0:16384/M-1);	SIMU1172
INICIALIZACAO: DO I=0 TO 16384/M-1;	SIMU1180
TIPO(I)='RAM';	SIMU1182
CARREGAMENTO(I)='1'B;	SIMU1184
TEMPO_LEITURA(I),TEMPO_ESCRITA(I)=0 ;	SIMU1186
END INICIALIZACAO;	SIMU1188
END;	SIMU1190
PUT EDIT((I, TIPO(I), CARREGAMENTO(I), TEMPO_LEITURA(I), TEMPO_ESCRITA(I)	
DO I=0 TO 16384/PAG-1))	
(COL(11), F(4), X(2), A(3), X(2), B(1), X(2), F(5), X(2), F(5));	
DO I=0 TO 16384/PAG-1;	
IF TIPO(I)='EQU' THEN IF TIPO(TEMPO_LEITURA(I))='EQU'	
TIPO(TEMPO_LEITURA(I))=' ' THEN DO;	
PUT SKIP LIST('ERRO DE EQUIVALENCIA DE PAGINAS(EQU=EQU / EQU= '));	
END; ELSE; ELSEF; END;	

```

/*****/SIMU1500
/* ***** P R O G R A M A S   P A R A   A   M E M O R I A   *****/SIMU1501
/*****/SIMU1502

```

```

CARREGAR:      /* CARREGAMENTO DOS PROGRAMAS E CONSTANTES NA MEMORIA      SIMU1504
                O PRIMEIRO CARTAO DE DADOS SERA UM CARTAO EM BRANCO.      SIMU1506
                VIRAO A SEGUIR OS BLOCOS DE PROGRAMAS.                    SIMU1508
                O ULTIMO CARTAO CONTERA DOIS NUMEROS POSITIVOS           SIMU1510
                QUAISQUER VINDO O MAIOR EM PRIMEIRO LUGAR.                */SIMU1512

```

```

GET SKIP LIST(INICIAL,FINAL);                                           SIMU1516
      /* LEITURA DE UM BLOCO DE PROGRAMA                               SIMU1518
        PRIMEIRO CARTAO DO BLOCO CONTEM DOIS NUMEROS:                   SIMU1520
        OS ENDEREÇOS LIMITES DO PROGRAMA NA MEMORIA.                   SIMU1522
        SEGUE-SE CARTOES INSTRUÇÃO E CONSTANTE                         */SIMU1524
IF INICIAL>FINAL THEN /* FIM DO CARREGAMENTO */ GOTO IMAGEM_DA_MEMORIA;SIMU1526
DO I=INICIAL/M TO FINAL/M;                                              SIMU1528
      IF CARREGAMENTO(I)='0'B THEN DO;                                    SIMU1530
PUT LIST('CARREGAMENTO PROIBIDO NO MODULO',I);                          SIMU1532
GOTO ENDE;                                                                SIMU1534
END;          FND;                                                        SIMU1536

```

```

LEITURA_DE_INSTRUcoes:/* FORMATO_DOS_CARTOES_DE_INSTRUCAO_E_CONSTANTE SIMU1538
                           COLUNA          FUNCAO          SIMU1542
                           1 A 8  INSTRUCAO OU CONSTANTE SOB FORMA BINARIA SIMU1544
                           9 A 72 COMENTARIO (OPCIONAL)      SIMU1546
                           73 A 80 NOME DA INSTRUCAO OU NUMERO DECIMAL SIMU1548
                               COMENTARIO E NOME SAO OPCIONAIS */SIMU1550
DO IMEM=INICIAL TO FINAL ;SIMU1552
  GET FOIT(MEM(IMEM),COMENTARIO,INSTRUCAO) SIMU1554
  (COLUMN(1),B(8),A(64),A(8)) ;SIMU1556
  PUT FOIT(MEM(IMEM),IMEM,INSTRUCAO,COMENTARIO) SIMU1558
  (COLUMN(23),B(8),COLUMN(35),F(3),COLUMN(40),A(8),COLUMN(50),A(64));SIMU1560
END LEITURA_DE_INSTRUcoes ;SIMU1562
PUT PAGE; SIMU1564
GOTO CARREGAR; SIMU1566

IMAGEM_DA_MEMORIA: SIMU1800
FIM_DA_PARTE_INICIAL='1'B;
PUT PAGE;
  GOTO MODULO_HLT;

```



```

/*****/SIMU2000
/* ***** M O D U L O   I N T E R R U P C A O *****/SIMU2001
/*****/SIMU2002
MODULO_INTERRUPT:
INTERUPCAO_UM:
    IF STOPPED<=1/* CONDICAO PARA HAVER INTERRUPCAO */ THEN DO;
    STOPPED=STOPPED+1;
        INTERRUPT=1B;
        VALOR_DA_INTERRUPTAO=1;
        /* GOTO MODULO_PCI; */ GOTO MODULO_PCIA;
        INT_PCI(1): REGI='11000000'B;          /* LAA */
        GOTO MODULO_DECODIFICACAO;
        INT_PCR_MM(1): REGB='00000000'B;
        REGA='00000000'B;
        GOTO MODULO_J_CA;
        INT_PCR_I(1): REGB='00000000'B;
        GOTO LABFL1;
    END;
INTERUPCAO_DOIS:
IF MOD(NUMERO_DE_ESTADOS,256)>=250 THEN DO;
    INTERRUPT=1B; IMPRIMA='1'B;
    VALOR_DA_INTERRUPTAO=2;
    INT_PCI(2): REGI='11001001'B; /* 6LBB */
    GOTO MODULO_DECODIFICACAO;
    INT_PCR_MM(2): REGB='00000000'B;
    REGA='00000001'B; /* 256 $ */
    GOTO MODULO_J_CA;
END;
IF NOME_DA_INSTRUCAO='HLT' THEN DO;
    IF STOPPED=5 THEN
        STOPPED=STOPPED+1;
        /* IF STOPPED=3 THEN */
        GOTO ENDE;
END;
/* INTERRUPT=0B; */ GOTO MODULO_PCIA;
/* FIM DO MODULO INTERRUPT */

```

```

/*****/SIMU4000
/* ***** D E C O D I F I C A C A O *****/SIMU4001
/*****/SIMU4002

```

```

MODULO_DECODIFICACAO: /* DECODIFICACAO_DA_INSIRUCAO_NO_REGISTRO_I */ SIMU4004
D7D6=(REGI&'11000000'B)/64; SIMU4008
D5D4D3=(REGI&'00111000'B)/8; SIMU4010
D2D1D0=(REGI&'00000111'B); SIMU4012
GOTO DECODIFICACAO_DE_D7D6(D7D6) ;SIMU4014
DECODIFICACAO_DE_D7D6(0): SIMU4016
GOTO DECODIFICACAO_DE_D2D1D0(D2D1D0); SIMU4018
DECODIFICACAO_DE_D2D1D0(2): SIMU4020
GOTO MODULO_RL_RC_RAL_R; SIMU4022
DECODIFICACAO_DE_D2D1D0(1): SIMU4024
DECODIFICACAO_DE_D2D1D0(0): SIMU4026
    IF D5D4D3=0 THEN GOTO MODULO_HLT; SIMU4028
    IF D5D4D3=7 THEN GOTO MODULO_INEXT; SIMU4030
    IF D2D1D0=1 THEN GOTO MODULO_DCR; SIMU4032
    GOTO MODULO_INR; SIMU4034
DECODIFICACAO_DE_D7D6(1): SIMU4036
    IF MOD(D2D1D0,2)=0 THEN GOTO MODULO_J_C; SIMU4038
    ELSE GOTO MODULO_I_0; SIMU4040
DECODIFICACAO_DE_D7D6(3): SIMU4042
    IF D2D1D0=7 THEN SIMU4044
        IF D5D4D3=7 THEN GOTO MODULO_HLT; SIMU4046
        ELSE GOTO MODULO_LRM; SIMU4048
    ELSE; SIMU4050
        IF D5D4D3=7 THEN GOTO MODULO_LMR; SIMU4052
        GOTO MODULO_LRR; SIMU4054
DECODIFICACAO_DE_D7D6(2): SIMU4056
    IF D2D1D0=7 THEN GOTO MODULO_ALU_M; SIMU4058
    GOTO MODULO_ALU_R; SIMU4060

```

```

/*****/SIMU4100
/* **** M L U U L L P C I ****/SIMU4101
/*****/SIMU4102
MODULO_PC1:
IF IMPRIMA='I'B|INTERRUPT=IE THEN
PC1 EDIT(NOME_DA_INSTRUCAO,(FCLND(1) DE I=0 TO 3),(REGINDEX(1) DE
I=0 TO 6),MCL(REGINDEX(5),64)*256+REGINDEX(6),OBSERVACAO,UCTEIO,
NUMERO_DE_ESTADOS)(CCL(28),A(6),4(X(1),F(1)),7(X(1),F(3)),X(1),F(5),
Z(X(1),A(8)),CCL(95),F(10));
INTERRUPT=OE;
OBSERVACAO=''; CCL(2)=';';
GLIC MODULO_INTERRUPT; MODULO_PC1a:
IMPRIMA=
(PC1(PC1)>TRACE_UN(1)&PC1(PC1)<TRACE_OFF(1))
PC1(PC1)>TRACE_UN(2)&PC1(PC1)<TRACE_OFF(2))
PC1(PC1)>TRACE_UN(3)&PC1(PC1)<TRACE_OFF(3))
PC1(PC1)>TRACE_UN(4)&PC1(PC1)<TRACE_OFF(4))
PC1(PC1)>TRACE_UN(5)&PC1(PC1)<TRACE_OFF(5))
PC1(PC1)>TRACE_UN(6)&PC1(PC1)<TRACE_OFF(6))
PC1(PC1)>TRACE_UN(7)&PC1(PC1)<TRACE_OFF(7))&INTERRUPT='O'B;
IF IMPRIMA='I'B THEN
PC1 SKIP EDIT(IPC,PC1(IPC))(CCL(20),F(1),X(1),F(5));
IF INTERRUPT=IB THEN GLIC INT_PC1(VALOR_DA_INTERRUPTCAU);
END_MEM=PC1(IPC);
MOD_MEM=END_MEM/M;
IF TIPC(MOD_MEM)=' ' THEN LC;
PC1 LIST('TENTATIVA DE LEITURA EM MODULO DE MEMORIA INEXISTENTE');
GLIC ENCL; ENCL;
IF TIPC(MOD_MEM)='EQU' THEN
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_LEITURA(MOD_MEM); ELSE
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+
TEMPO_LEITURA(TEMPO_LEITURA(MOD_MEM));
IF TIPC(MOD_MEM)='EQU' THEN REGI=MEM(END_MEM);
ELSE REGI=(MEM(PC1(END_MEM,M)+TEMPO_LEITURA(MOD_MEM)*M));
PC1(IPC)=MOD(PC1(IPC)+1,16384);
GLIC MODULO_DECOMPLICACAO;

```

```

SIMU4104
SIMU4106
SIMU4108
SIMU4110
SIMU4116
SIMU4117
SIMU4118
SIMU4120
SIMU4122
SIMU4124
SIMU4126
SIMU4128
SIMU4130
SIMU4132
SIMU4134
SIMU4138
SIMU4140
SIMU4142
SIMU4144
SIMU4146
SIMU4148
SIMU4150
SIMU4152
SIMU4154
SIMU4156
SIMU4158
SIMU4160
SIMU4162
SIMU4164
SIMU4166
SIMU4168

```

```

/*****/SIMU4200
/* ***** C I C L O   P C R *****/SIMU4201
/* ***** R E F E R E N C I A   I M E D I A T A *****/SIMU4202
/*****/SIMU4203
MODULO_I:                               SIMU4206
IF INTERRUPT =18 THEN GOTO INT_PCR_I(VALOR_DA_INTERRUPCAO);    SIMU4208
END_MEM=PC(IPC);                                                SIMU4210
MOD_MEM=END_MEM/M;                                              SIMU4212
IF TIPO(MOD_MEM)=' ' THEN DO;                                    SIMU4214
    PUT LIST('TENTATIVA DE LEITURA FM MODULO DE MEMORIA INEXISTENTE'); SIMU4216
    GOTO ENDE; END;                                             SIMU4218
IF TIPO(MOD_MEM)='FQU' THEN                                     SIMU4220
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_LEITURA(MOD_MEM); ELSE SIMU4222
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+
TEMPO_LEITURA(TEMPO_LEITURA(MOD_MEM));                       SIMU4226
IF TIPO(MOD_MEM)='FQU' THEN REGB=MEM(END_MEM);                 SIMU4228
ELSE REGB=(MEM(MOD(END_MEM,M))+TEMPO_LEITURA (MOD_MEM)*M));  SIMU4230
PC(IPC)=MOD(PC(IPC)+1,16384);                                   SIMU4232
OCTETO=REGB;                                                    SIMU4234
GOTO LABEL1;                                                    SIMU4236

```

```

/*****/SIMU4300
/* ***** C I C L O   P C R *****/SIMU4301
/* ***** R E F E R E N C I A   M E M O R I A *****/SIMU4302
/*****/SIMU4303
MODULO_M: SIMU4306
END_MEM=MOD(REGINDEX(5),64)*256+REGINDEX(6); SIMU4308
MOD_MEM=END_MEM/M; SIMU4310
IF TIPO(MOD_MEM)=' ' THEN DO; SIMU4312
    PUT LIST('TENTATIVA DE LEITURA EM MODULO DE MEMORIA INEXISTENTE'); SIMU4314
    GOTO ENDE; END; SIMU4316
IF TIPO(MOD_MEM)~='EQU' THEN SIMU4318
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_LEITURA(MOD_MEM); ELSE SIMU4320
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+ SIMU4322
    TEMPO_LEITURA(TEMPO_LEITURA(MOD_MEM)); SIMU4324
IF TIPO(MOD_MEM)~='EQU' THEN REGB=MEM(END_MEM); SIMU4326
ELSE REGB=(MEM(MOD(END_MEM,M)+TEMPO_LEITURA(MOD_MEM)*M)); SIMU4328
OCTETO=REGB; SIMU4330
GOTO LABEL2; SIMU4332

```

```

/*****/SIMU4400
/* ***** C I C L O   P C R *****/SIMU4401
/* ***** R E F E R E N C I A   M E M   M E M *****/SIMU4402
/*****/SIMU4403
MODULO_MM: SIMU4406
IF INTERRUPT=18 THEN GOTO INT_PCR_MM(VALOR_DA_INTERRUPCAO); SIMU4408
END_MEM=PC(IPC); SIMU4410
MOD_MEM=END_MEM/M; SIMU4412
IF TIPO(MOD_MEM)=' ' THEN DO; SIMU4414
    PUT LIST('TENTATIVA DE LEITURA EM MODULO DE MEMORIA INEXISTENTE'); SIMU4416
    GOTO ENDE; END; SIMU4418
IF TIPO(MOD_MEM)='EQU' THEN SIMU4420
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_LEITURA(MOD_MEM); ELSE SIMU4422
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+
    TEMPO_LEITURA(TEMPO_LEITURA(MOD_MEM)); SIMU4426
IF TIPO(MOD_MEM)='EQU' THEN REGB=MEM(END_MEM); SIMU4428
ELSE REGB=(MEM(MOD(END_MEM,M)+TEMPO_LEITURA(MOD_MEM)*M)); SIMU4430
PC(IPC)=MOD(PC(IPC)+1,16384); SIMU4432
END_MEM=PC(IPC); SIMU4434
MOD_MEM=END_MEM/M; SIMU4436
IF TIPO(MOD_MEM)=' ' THEN DO ; SIMU4438
    PUT LIST('TENTATIVA DE LEITURA EM MODULO DE MEMORIA INEXISTENTE'); SIMU4440
    GOTO ENDE; END; SIMU4442
IF TIPO(MOD_MEM)='EQU' THEN SIMU4444
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_LEITURA(MOD_MEM); ELSE SIMU4446
    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+
    TEMPO_LEITURA(TEMPO_LEITURA(MOD_MEM)); SIMU4450
IF TIPO(MOD_MEM)='EQU' THEN REGA=MEM(END_MEM); SIMU4452
ELSE REGA=(MEM(MOD(END_MEM,M)+TEMPO_LEITURA(MOD_MEM)*M)); SIMU4454
PC(IPC)=MOD(PC(IPC)+1,16384); SIMU4456
GOTO MODULO_J_Ca; SIMU4458

```

```

/*****/SIMU4500
/* ***** C I C L O   P C W *****/SIMU4501
/*****/SIMU4502
MODULO_PCW: SIMU4504
END_MEM=MOD(REGINDEX(5),64)*256+REGINDEX(6); SIMU4506
MOD_MEM=END_MEM/M; SIMU4508
IF TIPO(MOD_MEM)=' ' THEN DO ; SIMU4510
    PUT LIST('TENTATIVA DE ESCRITA EM MODULO DE MEMORIA INEXISTENTE'); SIMU4512
GOTO ENDE; END; SIMU4514
IF TIPO(PAG_MEM)≠'EQU' THEN
    IF TIPO(PAG_MEM)='ROM' THEN OBSERVACAO='VIOL MEM';
        ELSE DO; MEM(END_MEM)=REGB;
            NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+TEMPO_ESCRITA(PAG_MEM);END;
        ELSE
            IF TIPO(TEMPO_LEITURA(PAG_MEM))='ROM' THEN OBSERVACAO='VIOL MEM';
                ELSE DO;
                    MEM(MOD(END_MEM,M)+TEMPO_LEITURA(PAG_MEM)*M)=REGB;
                    NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+
                        TEMPO_ESCRITA(TEMPO_LEITURA(PAG_MEM)); END;
GOTO MODULO_PCI; SIMU4536

/*****/SIMU4600
/* ***** M O D U L O   P C C *****/SIMU4601
/*****/SIMU4602
MODULO_PCC: SIMU4604
OUT_LATCH=REGINDEX(0); SIMU4606
VALOR_PERIFERICO=FLOOR(D5D4D3/2)*10+MOD(D5D4D3,2)*4+(D2D1D0-1)/2; SIMU4608
GOTO MODULO_I_0a; SIMU4610

```

```

/*****/SIMU5000
/* **** L O A D   R E G   R E G *****/SIMU5001
/*****/SIMU5002
MODULO_LRR:                                     SIMU5004
NOME_DA_INSTRUCAO='L' || NOME_DO_REGISTRO(D5D4D3) || NOME_DO_REGISTRO(D2D1D0); SIMU5008
0);                                             SIMU5010
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5;       SIMU5014
REGINDEX(D5D4D3)=REGINDEX(D2D1D0);           SIMU5016
GOTO MODULO_PCI;                             SIMU5018

/*****/SIMU5050
/* **** L O A D   R E G   M E M O R I A *****/SIMU5051
/*****/SIMU5052
MODULO_LRM:                                     SIMU5054
LABEL2=MODULO_LRM@; GOTO MODULO_M;            SIMU5056
MODULO_LRM@:                                   SIMU5058
NOME_DA_INSTRUCAO='L' || NOME_DO_REGISTRO(D5D4D3) || 'M'; SIMU5062
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+8;       SIMU5066
REGINDEX(D5D4D3)=REGB;                       SIMU5068
GOTO MODULO_PCI;                             SIMU5070

/*****/SIMU5100
/* **** L O A D   M E M O R I A   R E G I S T R O *****/SIMU5101
/*****/SIMU5102
MODULO_LMR:                                     SIMU5104
NOME_DA_INSTRUCAO='LM' || NOME_DO_REGISTRO(D2D1D0); SIMU5108
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+7;       SIMU5110
REGB=REGINDEX(D2D1D0);                       SIMU5112
GOTO MODULO_PCW;                             SIMU5114

```



```

/*****/SIMU5150
/* ***** L O A D   I M E D I A T O *****/SIMU5151
/*****/SIMU5152

```

```

DECODIFICACAO_DE_D2D1D0(6): MODULO_LR_MI:          SIMU5154
LABEL1=MODULO_LR_MI@; GOTO MODULO_I;                SIMU5156
MODULO_LR_MI@:                                       SIMU5158
IF D5D4D3=7 THEN DO;                                SIMU5160
  MODULO_LMI:                                       SIMU5162
  NOME_DA_INSTRUCAO='LMI';                          SIMU5164
  NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+9;            SIMU5166
  GOTO MODULO_PCW;   END;                           SIMU5168
MODULO_LRI:                                         SIMU5170
NOME_DA_INSTRUCAO='L' || NOME_DO_REGISTRO(D5D4D3) || 'I'; SIMU5172
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+8;              SIMU5174
REGINDEX(D5D4D3)=REGB;                             SIMU5176
GOTO MODULO_PCI;                                    SIMU5178

```

```

/*****/SIMU5200
/* ***** M O D U L O   I N C R E M E N T O   R E G *****/SIMU5201
/*****/SIMU5202
MODULO_INR: SIMU5204
NOME_DA_INSTRUCAO='IN' || NOME_DO_REGISTRO(D5D4D3); SIMU5206
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5; SIMU5208
OVERFLOW=CARRY; SIMU5210
REGINDEX(D5D4D3),REGTESTE=MOD(REGINDEX(D5D4D3)+1,256); SIMU5212
GOTO MODULO_C_Z_S_P; SIMU5214

/*****/SIMU5250
/* ***** M O D U L O   D E C R E M E N T O   R E G *****/SIMU5251
/*****/SIMU5252
MODULO_DCR: SIMU5254
NOME_DA_INSTRUCAO='DC' || NOME_DO_REGISTRO(D5D4D3); SIMU5256
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5; SIMU5258
UNDERFLOW=CARRY; SIMU5260
REGINDEX(D5D4D3),REGTESTE=MOD(REGINDEX(D5D4D3)-1,256); SIMU5262
GOTO MODULO_C_Z_S_P; SIMU5264

```

```

/*****/SIMU5300
/* ***** M O D U L O   A R I T M E T I C O   R E G I S T R O*/SIMU5301
/*****/SIMU5302
MODULO_ALU_R:                                SIMU5304
TIPO_DE_REFERENCIA=NOME_DO_REGISTRO(D2D1D0);    SIMU5308
      REGB=REGINDEX(D2D1D0);                    SIMU5312
      NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5;      SIMU5314
      GOTO MODULO_ALU;                            SIMU5316

/*****/SIMU5350
/* ***** M O D U L O   A R I T M E T I C O   M E M *****/SIMU5351
/*****/SIMU5352
MODULO_ALU_M:                                SIMU5354
TIPO_DE_REFERENCIA='M'; NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+8;    SIMU5356
LABEL2=MODULO_ALU; GOTO MODULO_M;              SIMU5358

/*****/SIMU5400
/* ***** M O D U L O   A L U   I M E D I A T O *****/SIMU5401
/*****/SIMU5402

DECODIFICACAO_DE_D2D1D0(4): MODULO_ALU_I:      SIMU5404
TIPO_DE_REFERENCIA='I'; NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+8;    SIMU5406
LABEL1=MODULO_ALU; GOTO MODULO_I;              SIMU5408

```

```

/*****/SIMU5450
/* ***** M O D U L O   D E C O D I F I C A C A O   A L U *****/SIMU5451
/*****/SIMU5452
MODULO_ALU:                               SIMU5452
    IF D5D4D3=0 THEN GOTO MODULO_AD;       SIMU5456
    IF D5D4D3=1 THEN GOTO MODULO_AC;       SIMU5458
    IF D5D4D3=2 THEN GOTO MODULO_SU;       SIMU5460
    IF D5D4D3=3 THEN GOTO MODULO_SB;       SIMU5462
    IF D5D4D3=4 THEN GOTO MODULO_ND;       SIMU5458
    IF D5D4D3=5 THEN GOTO MODULO_XR;       SIMU5466
    IF D5D4D3=6 THEN GOTO MODULO_OR;       SIMU5468
                                GOTO MODULO_CP; SIMU5470

```

```

/*****/SIMU5500
/* ***** M O D U L O   A D *****/SIMU5501
/*****/SIMU5502
MODULO_AD: SIMU5504
NOME_DA_INSTRUCAO='AD' || TIPO_DE_REFERENCIA; SIMU5506
MODULO_AD@: SIMU5508
IF ACUMULADOR(1)+REGB>=256 THEN OVERFLOW=1B; SIMU5510
REGTSTE,ACUMULADOR(1)=MOD(ACUMULADOR(1)+REGB,256); SIMU5512
GOTO MODULO_C_Z_S_P; SIMU5514

/*****/SIMU5550
/* ***** M O D U L O   A C *****/SIMU5551
/*****/SIMU5552
MODULO_AC: SIMU5554
NOME_DA_INSTRUCAO='AC' || TIPO_DE_REFERENCIA; SIMU5556
IF REGB+CARRY(1)=256 THEN OVERFLOW=1B; SIMU5558
REGB=MOD(REGB+CARRY(1),256); SIMU5560
GOTO MODULO_AD@; SIMU5562
/*****/SIMU5600
/* ***** M O D U L O   S U *****/SIMU5601
/*****/SIMU5602
MODULO_SU: SIMU5604
NOME_DA_INSTRUCAO='SU' || TIPO_DE_REFERENCIA; SIMU5606
REGB=256-REGB; SIMU61/,
GOTO MODULO_AD@; SIMU5610
/*****/SIMU5650
/* ***** M O D U L O   S B *****/SIMU5651
/*****/SIMU5652
MODULO_SB: SIMU5604
NOME_DA_INSTRUCAO='SB' || TIPO_DE_REFERENCIA; SIMU5606
IF REGB+BORROW(1)=256 THEN UNDERFLOW=1B; SIMU5608
REGB=256- MOD(REGB+CARRY(1),256); SIMU5610
GOTO MODULO_AD@; SIMU5612

```

```

/* *****/SIMU5650
/* ***** M O D U L O   N D *****/SIMU5651
/******/SIMU5652
MODULO_ND:
NOME_DA_INSTRUCAO='ND' || TIPO_DE_REFERENCIA;
REGTESTE, ACUMULADOR(1)=REGB&ACUMULADOR(1);
GOTO MODULO_C_Z_S_P;

/******/SIMU5700
/* ***** M O D U L O   X R *****/SIMU5701
/******/SIMU5702
MODULO_XR:
NOME_DA_INSTRUCAO='XR' || TIPO_DE_REFERENCIA;
REGTESTE, ACUMULADOR(1)=~REGB&ACUMULADOR(1) | REGB&~ACUMULADOR(1);
GOTO MODULO_C_Z_S_P;

/******/SIMU5750
/* ***** M O D U L O   O R *****/SIMU5751
/******/SIMU5752
MODULO_OR:
NOME_DA_INSTRUCAO='OR' || TIPO_DE_REFERENCIA;
REGTESTE, ACUMULADOR(1)=REGB|ACUMULADOR(1);
GOTO MODULO_C_Z_S_P;

/******/SIMU5800
/* ***** M O D U L O   C P *****/SIMU5801
/******/SIMU5802
MODULO_CP:
NOME_DA_INSTRUCAO='CP' || TIPO_DE_REFERENCIA;
IF ACUMULADOR(1)<REGB THEN UNDERFLOW=1B;
REGTESTE=MOD(ACUMULADOR(1)+(256-REGB),256);
GOTO MODULO_C_Z_S_P;

```

```

/*****/SIMU5900
/* ***** R O T A C A O D O A C U M U L A D O R *****/SIMU5901
/*****/SIMU5902
MODULO_RL_RC_RAL_R: SIMU5904
NUMERO_DE_FSTADOS=NUMERO_DE_FSTADOS+5; SIMU5906
CARRYAUX=CARRY; CARRY=0B; GOTO LABEL_ROTATE(MOD(D5D4D3,4)); SIMU5908
LABEL_ROTATE(0): NOME_DA_INSTRUCAO='RLC'; GOTO MODULO_RLC@; SIMU5910
LABEL_ROTATE(1): NOME_DA_INSTRUCAO='RRC'; GOTO MODULO_RRC@; SIMU5912
LABEL_ROTATE(2): NOME_DA_INSTRUCAO='RAL'; GOTO MODULO_RAL@; SIMU5914
LABEL_ROTATE(3): NOME_DA_INSTRUCAO='RAR'; GOTO MODULO_RAR@; SIMU5916
MODULO_RLC@: MODULO_RAL@: SIMU5918
IF ACUMULADOR(1)>127 THEN CARRY=1B; SIMU5920
ACUMULADOR= MOD(ACUMULADOR(1)*2,256)+ SIMU5922
    CARRY*(D5D4D3=0)+ CARRYAUX*(D5D4D3=2); SIMU5924
GOTO MODULO_PCI; SIMU5926
MODULO_RRC@: MODULO_RAR@: SIMU5928
IF MOD(ACUMULADOR(1),2)=1 THEN CARRY=1B; SIMU5930
ACUMULADOR=ACUMULADOR/2+ SIMU5932
    128*(CARRY*(D5D4D3=1)+CARRYAUX*(D5D4D3=3)); SIMU5934
GOTO MODULO_PCI; SIMU5936

```

```

/*****/SIMU6000
/* ***** M O D U L O   J U M P   C A L L *****/SIMU6001
/*****/SIMU6002
MODULO_J_C: GOTO MODULO_MM; SIMU6004
MODULO_J_Ca: SIMU6006
IF D5D4D3>3 THEN DO; D5D4D3=D5D4D3-4;CONDICAO=1B;TRUE_FALSE='T';END; SIMU6008
ELSE DO; CONDICAO=0B;TRUE_FALSE='F';END; SIMU6010
GOTO LABEL_JUMP_CALL(D2D1D0/2); SIMU6012
LABEL_JUMP_CALL(3): NOME_DA_INSTRUCAO='CAL'; GOTO MODULO_CAL; SIMU6014
LABEL_JUMP_CALL(2): NOME_DA_INSTRUCAO='JMP'; GOTO MODULO_JMP; SIMU6016
LABEL_JUMP_CALL(1): SIMU6018
NOME_DA_INSTRUCAO='C' || TRUE_FALSE || NOME_DO_FF(D5D4D3); SIMU6020
IF CONDICAO=FFCOND(D5D4D3) THEN DO; SIMU6022
OBSERVACAO=' PULA '; GOTO MODULO_CAL; END; SIMU6024
ELSE DO; SIMU6026
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+9; SIMU6028
OBSERVACAO='NAO PULA'; GOTO MODULO_PCI; END; SIMU6030
LABEL_JUMP_CALL(0): SIMU6032
NOME_DA_INSTRUCAO='J' || TRUE_FALSE || NOME_DO_FF(D5D4D3); SIMU6034
IF CONDICAO=FFCOND(D5D4D3) THEN DO; SIMU6036
OBSERVACAO=' PULA '; GOTO MODULO_JMP; END; SIMU6038
ELSE DO; SIMU6040
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+9; SIMU6042
OBSERVACAO='NAO PULA'; GOTO MODULO_PCI; END; SIMU6044
MODULO_CAL: IPC=MOD(IPC+1,8); SIMU6046
MODULO_JMP: PC(IPC)=MOD(REGA*256+REGR,16384) ; SIMU6048
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+11; GOTO MODULO_PCI; SIMU6050

```



```

/*****/SIMU6100
/* ***** M O D U L O   R E T U R N *****/SIMU6101
/*****/SIMU6102

```

```

DECODIFICACAO_DE_D2D1D0(7): MODULO_RET:          SIMU6104
NOME_DA_INSTRUCAO='RET';                          SIMU6106
    MODULO_RET@:                                    SIMU6108
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5; IPC=MOD(IPC-1,8); SIMU6110
GOTO MODULO_PCI;                                    SIMU6112
/*****/SIMU6150
/* ***** M O D U L O   R E T U R N   C O N D I C I O N A L *****/SIMU6151
/*****/SIMU6152

```

```

DECODIFICACAO_DE_D2D1D0(3): MODULO_RF_TC:          SIMU6154
IF D5D4D3>3 THEN DO; D5D4D3=D5D4D3-4; TRUE_FALSE='T'; CONDICAO=1B; END;SIMU6156
    ELSE DO;                                         TRUE_FALSE='F'; CONDICAO=0B; END;SIMU6158
NOME_DA_INSTRUCAO='R' || TRUE_FALSE || NOME_DO_FF(D5D4D3); SIMU6160
IF FFCOND(D5D4D3)=CONDICAO THEN DO; OBSERVACAO=' PULA  '; SIMU6162
    GOTO MODULO_RET@; END;                          SIMU6164
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+3; OBSERVACAO='NAO PULA'; SIMU6166
GOTO MODULO_PCI;                                    SIMU6168

```

```

/*****/SIMU6200
/* ***** M O D U L O   R E S T A R T *****/SIMU6201
/*****/SIMU6202

```

```

DECODIFICACAO_DE_D2D1D0(5): MODULO_RST:            SIMU6204
NOME_DA_INSTRUCAO='RST' || '_' ||                 SIMU6206
SUBSTR(CHAR(D5D4D3*8),LENGTH(CHAR(D5D4D3*8))-1); SIMU6208
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5; IPC=MOD(IPC+1,8); SIMU6210
PC(IPC),REGB=D5D4D3*8; GOTO MODULO_PCI;            SIMU6212

```

```

/*****
/* ***** M O D U L O   I N P U T   O U T P U T *****
/*****
MODULO_I_0:
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+8; GOTO MODULO_PCC;
MODULO_I_0a:
IF D5D4D3<=1 THEN DO;
    FF LATCH=CARRY(1)*1000B+PRTY(1)*100B+ZERO(1)*10B+SIGN(1) ;
NOME_DA_INSTRUCAO='INP' || '_' || SUBSTR(VALOR_PERIFERICO,4,2);
    GOTO MODULO_INP(VALOR_PERIFERICO);
    END;
NOME_DA_INSTRUCAO='OUT' || '_' || SUBSTR(VALOR_PERIFERICO,4,2);
    GOTO MODULO_OUT(VALOR_PERIFERICO);

```

```

MDULL_INP(1):
MDULL_INP(0):
MDULL_INP(2):
MDULL_INP(3):
MDULL_INP(4):
MDULL_INP(5):
ACUMULADUR=255;
GOTO MDULL_PUL;
MDULL_INP(6):
DEIN;
LN CRRK INDEX=1;
GOTO VALCK_UL_BIT(5) INIT(0,0,1,1,1);
INDEX=MUL(INDEX+1,5) +1;
ACUMULADUR(1)=VALCK_UL_BIT(INDEX);
END;
GOTO MDULL_PUL;
MDULL_INP(7):
ACUMULADUR(1) ,VALCK_DE_ENTRADA(0)=MUL(LFLL_DE_ENTRADA(0)+1,8);
GOTO MDULL_PUL;

```

```

      . MODULO_OUT(10):
MODULO_OUT(11):
MODULO_OUT(12):
MODULO_OUT(13):
MODULO_OUT(14):
MODULO_OUT(15):
MODULO_OUT(16):
MODULO_OUT(17):
MODULO_OUT(20):
MODULO_OUT(21):
MODULO_OUT(22):
MODULO_OUT(23):
MODULO_OUT(24):
MODULO_OUT(25):
MODULO_OUT(26):
MODULO_OUT(27):
MODULO_OUT(30):
MODULO_OUT(31):
MODULO_OUT(32):
MODULO_OUT(33):
MODULO_OUT(34):
MODULO_OUT(35):
GOTO MODULO_PCI;
MODULO_OUT(36):
PUT EDIT((MEM(J) DO J=OUT_LATCH*64 TO OUT_LATCH*64+63))
(COL(30),B(8),COL(40),B(8),COL(50),B(8),COL(60),B(8),COL(70),B(8),
COL(80),B(8),COL(90),B(8),COL(100),B(8));
GOTO MODULO_PCI;
MODULO_OUT(37):
      NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS +0;
GUARDA_ACUMULADOR=ACUMULADOR(1);
      GOTO MODULO_PCI;

```

```

/*****/SIMU9850
/* ***** M O D U L O   H A L T *****/SIMU9851
/*****/SIMU9852
MODULO_HLT:                                     SIMU9854
NOME_DA_INSTRUCAO='HLT';                         SIMU9856
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+4;           SIMU9858
GOTO MODULO_PCI;                                SIMU9860

/*****/SIMU9900
/* ***** I N S T R U C A O   I N E X I S T E N T E *****/SIMU9901
/*****/SIMU9902
MODULO_INEXT:                                    SIMU9904
NUMERO_DE_ESTADOS=NUMERO_DE_ESTADOS+5;           SIMU9906
IF D2D1D0 THEN DO;NOME_DA_INSTRUCAO='SET101';REGTESTE=00000000B;END;   SIMU9908
      ELSE DO;NOME_DA_INSTRUCAO='SET010';REGTESTE=11111110B;END;       SIMU9910
GOTO MODULO_PCI;                                SIMU9912

```

```

/*****/SIMU9950
/* ***** M O D U L O   F F   C O N D I C A O *****/SIMU9951
/*****/SIMU9952

```

```

MODULO_C_Z_S_P:                                SIMU9954
CONTADOR=0;CAPRY=OVERFLOW;UNDERFLOW,OVERFLOW=0B;    SIMU9956
IF REGTESTE=0 THEN DO;ZERO,PRTY=1B;SYGN=0B;GOTO MODULO_PCI;END;    SIMU9958
    ELSE ZERO=0B;                                SIMU9960
IF REGTESTE>127 THEN SYGN=1B; ELSE SYGN=0B;    SIMU9964
DO NNN=7 TO 0 BY -1;                                SIMU9966
    IF REGTESTE>2**NNN-1 THEN DO;CONTADOR=CONTADOR+1;    SIMU9968
        REGTESTE=REGTESTE-2**NNN; END;    SIMU9970
    END;                                SIMU9972
IF MOD(CONTADOR,2)=0 THEN PRTY=1B; ELSE PRTY=0B;    SIMU9974
    GOTO MODULO_PCI;                                SIMU9976

```

```

ENDE:
FREE IMAGEM_MEMORIA;
END NOVO_PROGRAMA;      GOTO NOVO_PROGRAMA;
GRAN_FINAL: END;

```

C	REM	1	0	0	
			00000110	0	LAI
			00000001	1	
			00110000	2	INL
			10000000	3	ADA
			01010000	4	JFS
			00000011	5	
			00000000	6	
			11111000	7	LMA
			10000111	8	ADM
			10111101	9	CPH
			01101010	10	CTZ
			00000000	11	
			01111111	12	
			01100010	13	CTC
			00001001	14	
			11111111	15	
			01111100	16	JMP
			11111001	17	
			11111111	18	

00000110	381	LAI
00000000	382	
01111011	383	CUT 35

00110001	128	DCL
11000111	129	LAM
00110000	130	INL
10110111	131	GRM
00010010	132	RAL
00001010	133	RRC
11111000	134	LMA
00000101	135	RST 0
00111111	136	RET
10101000	137	XRA
11110000	138	LLA
10101000	139	XRA
11111000	140	LMA
00110000	141	INL
00000110	142	LAI
00000011	143	
10111110	144	CPL
00101011	145	RTZ
01001100	146	JMP
00001011	147	
00111111	148	

		HLT	0	0	0	0	0	0	0	0	0	0	0	0	4
		LAA	0	0	0	0	0	0	0	0	0	0	0	0	9
		LAA	0	0	0	0	0	0	0	0	0	0	0	0	14
0	0	LAI	0	0	0	0	1	0	0	0	0	0	0	0	22
0	2	INL	0	0	0	0	1	0	0	0	0	1	1	1	27
0	3	ADA	0	0	0	0	2	0	0	0	0	1	1	1	32
0	4	JFS	0	0	0	0	2	0	0	0	0	1	1	1	43
0	3	ADA	0	0	0	0	4	0	0	0	0	1	1	1	48
0	4	JFS	0	0	0	0	4	0	0	0	0	1	1	1	59
0	3	ADA	0	0	0	0	8	0	0	0	0	1	1	1	64
0	4	JFS	0	0	0	0	8	0	0	0	0	1	1	1	75
0	3	ADA	0	0	0	0	16	0	0	0	0	1	1	1	80
0	4	JFS	0	0	0	0	16	0	0	0	0	1	1	1	91
0	3	ADA	0	0	0	0	32	0	0	0	0	1	1	1	96
0	4	JFS	0	0	0	0	32	0	0	0	0	1	1	1	107
0	3	ADA	0	0	0	0	64	0	0	0	0	1	1	1	112
0	4	JFS	0	0	0	0	64	0	0	0	0	1	1	1	123
0	3	ADA	0	0	1	0	128	0	0	0	0	1	1	1	128

1

PULA

PULA

PULA

PULA

PULA

PULA

0	4	JFS	0	0	1	0	128	0	0	0	0	0	1	1	NAO PULA	137
0	7	LMA	0	0	1	0	128	0	0	0	0	0	1	1	VIOL MEM	144
0	8	ADM	0	0	1	1	129	0	0	0	0	0	1	1		152
0	9	CPH	0	0	1	1	129	0	0	0	0	0	1	1		157
0	10	CTZ	0	0	1	1	129	0	0	0	0	0	1	1	NAO PULA	166
0	13	CTC	0	0	1	1	129	0	0	0	0	0	1	1	NAO PULA	175
0	16	JMP	0	0	1	1	129	0	0	0	0	0	1	1		186
0	16381	CFS	0	0	1	1	129	0	0	0	0	0	1	1	NAO PULA	195
0	0	LAI	0	0	1	1	1	0	0	0	0	0	1	1		203
0	2	INL	0	0	0	0	1	0	0	0	0	0	2	2		208
0	3	ABA	0	0	0	0	2	0	0	0	0	0	2	2		213
0	4	JFS	0	0	0	0	2	0	0	0	0	0	2	2	PULA	224
0	3	ADA	0	0	0	0	4	0	0	0	0	0	2	2		229
0	4	JFS	0	0	0	0	4	0	0	0	0	0	2	2	PULA	240
0	3	ADA	0	0	0	0	8	0	0	0	0	0	2	2		245
0	4	JFS	0	0	0	0	8	0	0	0	0	0	2	2	PULA	256
0	3	ADA	0	0	0	0	16	0	0	0	0	0	2	2		261
0	4	JFS	0	0	0	0	16	0	0	0	0	0	2	2	PULA	272
0	3	ADA	0	0	0	0	32	0	0	0	0	0	2	2		277
0	4	JFS	0	0	0	0	32	0	0	0	0	0	2	2	PULA	288
0	3	ADA	0	0	0	0	64	0	0	0	0	0	2	2		293
0	4	JFS	0	0	0	0	64	0	0	0	0	0	2	2	PULA	304
0	3	ADA	0	0	1	0	128	0	0	0	0	0	2	2		309
0	4	JFS	0	0	1	0	128	0	0	0	0	0	2	2	NAO PULA	318
0	7	LMA	0	0	1	0	128	0	0	0	0	0	2	2	VIOL MEM	325
0	8	ADM	0	0	1	0	176	0	0	0	0	0	2	2		333
0	9	CPH	0	0	1	0	176	0	0	0	0	0	2	2		338
0	10	CTZ	0	0	1	0	176	0	0	0	0	0	2	2	NAO PULA	347
0	13	CTC	0	0	1	0	176	0	0	0	0	0	2	2	NAO PULA	356

1

1

48

0	16	JMP	0	0	1	0	176	C	0	0	0	0	2	2		367
0	16381	CFS	0	0	1	0	176	0	0	0	0	0	2	2	NAD PULA	376
0	C	LAI	0	0	1	0	1	C	0	0	0	0	2	2		384

1

0	2	INL	0	0	0	1	1	0	0	0	0	0	0	3	3		389
0	3	ACA	0	0	0	0	2	0	0	0	0	0	0	3	3		394
0	4	JFS	0	0	0	0	2	0	0	0	0	0	0	3	3	PULA	405
0	3	ACA	0	0	0	0	4	0	0	0	0	0	0	3	3		410
0	4	JFS	0	0	0	0	4	0	0	0	0	0	0	3	3	PULA	421
0	3	ACA	0	0	0	0	8	0	0	0	0	0	0	3	3		426
0	4	JFS	0	0	0	0	8	0	0	0	0	0	0	3	3	PULA	437
0	3	ACA	0	0	0	0	16	0	0	0	0	0	0	3	3		442
0	4	JFS	0	0	0	0	16	0	0	0	0	0	0	3	3	PULA	453
0	3	ACA	0	0	0	0	32	0	0	0	0	0	0	3	3		458

IPC	PC	INST	C	Z	S	P	A	B	C	D	E	H	L	M		ESTADOS
0	4	JFS	0	0	0	0	32	0	0	0	0	0	3	3	PULA	469
0	3	ACA	0	0	0	0	64	0	0	0	0	0	3	3		474
0	4	JFS	0	0	0	0	64	0	0	0	0	0	3	3	PULA	485
0	3	ADA	0	0	1	0	128	0	0	0	0	0	3	3		490
0	4	JFS	0	0	1	0	128	0	0	0	0	0	3	3	NAO PULA	499
0	7	LMA	0	0	1	0	128	0	0	0	0	0	3	3	VIOL MEM	506
		LBB	0	0	1	0	128	0	0	0	0	0	3	3		511
		LBB	0	0	1	0	128	0	0	0	0	0	3	3		516
0	8	ADM	1	1	0	1	0	0	0	0	0	0	3	3		524
0	9	CPH	0	1	0	1	0	0	0	0	0	0	3	3		529
0	10	CTZ	0	1	0	1	0	0	0	0	0	0	3	3	PULA	540
1	16128	INP_ 3	0	1	0	1	255	0	0	0	0	0	3	3		548
1	16129	LLA	0	1	0	1	255	0	0	0	0	0	255	255		553
1	16130	SLA	1	1	0	1	0	0	0	0	0	0	255	255		558
1	16131	JMP	1	1	0	1	0	0	0	0	0	0	255	255		569
1	12359	HLT	1	1	0	1	0	0	0	0	0	0	255	255		573
1	12360	HLT	1	1	0	1	0	0	0	0	0	0	255	255		577
1	12361	HLT	1	1	0	1	0	0	0	0	0	0	255	255		581

128